Spring 2010

Topics in access, storage, and sensor networks

Swapnil Bhatia
University of New Hampshire, Durham

Topics in Access, Storage, and Sensor Networks

BY

Swapnil Bhatia

Bachelor of Computer Engineering, Mumbai University (1999)

DISSERTATION

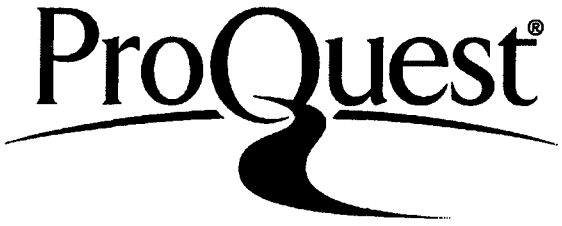
Submitted to the University of New Hampshire
in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

in

Computer Science

May 2010



ProQuest LLC
789 East Eisenhower Parkway
P.O. Box 1346
Ann Arbor, MI 48106-1346

This dissertation has been examined and approved.

Dissertation Director, Radim Bartoš, Associate Professor

Computer Science

Michael J. Carter, Associate Professor

Electrical and Computer Engineering

Michel Charpentier, Associate Professor

Computer Science

Pilar de la Torre, Professor (Emerita)

Computer Science

Xiaozhou Li, Research Scientist

Hewlett Packard Laboratories

Elizabeth Varki, Associate Professor

Computer Science

January 22nd, 2010

Date

Dedication

This dissertation is dedicated to my family and my teachers.

Acknowledgments

The work described in this dissertation was funded in part by grants from the Cisco University Research Program, the Office of Naval Research, and the CEPS Teaching Award and support from the UNH InterOperability Laboratory. I am grateful for this support.

It is a pleasure to acknowledge the contribution and support of my colleagues and my teachers: to a large extent this is a result of their ideas, intuition, questions, and effort. I claim full credit for all errors, omissions, shortcomings, and hideousness of presentation.

The work described in Chapter 6 is a result of joint work with fellow graduate student Chaitanya Godsay. This work was performed at and supported by the UNH InterOperability Laboratory (IOL). I thank Barry Reinhold and Steven Fulton from the IOL for their managerial leadership which led to and supported this work. Graduate students Vrushali Ashtaputre, Ehteshamul Haque, and Dmitri Garbuzov helped jumpstart the weekly reading of papers on access networks: their energy and enthusiasm propelled to completion what is now Chapter 3. I also appreciate Dmitri Garbuzov's enthusiasm in our study of the IPACT protocol, reported in Chapter 4. My work on access networks benefited tremendously from my discussions with Dr. Glen Kramer (Univ. California, Davis and Teknovus Inc.), Wael Diab and Russ Gyurek (Cisco Systems Inc.), and Eric Lynskey (IOL). The work described in Chapter 8 is joint work with Prof. Michel Charpentier and graduate student Kevin Ma, who also collaborated on the work described in Chapter 11. Profs. Charpentier and Bartoš helped develop the ideas and their presentation in Sec. 8.4.

The initial study of population protocols (Chapter 9) was suggested by and carried out with Prof. Pilar de la Torre. The results reported in Chapter 9 were motivated by Prof. Michel Charpentier's work (and his CS900 talk) on self-similar algorithms. I thank Justin Greenough (graduate student in the Department of Mathematics and Statistics) and graduate student Jeff Laird for their encouragement and for allowing me to subject them

to the ideas that became Chapter 10. The work described in Chapter 12 and Chapter 13 is part of a joint project led by Prof. Elizabeth Varki and graduate student Mingju Li and Dr. Arif Merchant of Hewlett Packard Laboratories. Prof. Varki shaped the presentation of the ideas in Secs. 12.1 to 12.5. I benefited tremendously from Dr. Merchant's early feedback on Chapter 12. All of the work described in this dissertation was carried out under the gentle guidance of and with abundant contributions from my advisor Prof. Radim Bartoš; this dissertation is an attempt to thank him warmly.

I thank my collaborators for their work and their delightful company. I thank Arun Gandhi for being an excellent and instructive first collaborator. It was a pleasure to work with Mingju Li and I thank her for always welcoming my uncountable list of questions. I thank Kevin for generously allowing me to participate in his work and for teaching me about streaming video and mobile devices. I thank Justin for teaching me many things and for sharing some of his mathematics with me. I thank Andy Foulks, Justin Greenough, Jeff Laird, and Kevin Ma for many fascinating discussions and their invaluable friendship. I learned a lot from them and they made my stay at UNH a lot of fun. I especially thank Jeff and Justin for always entertaining my questions and giving them careful consideration.

I thank the members of my committee for their careful reading of the dissertation and their insightful questions and feedback. I thank Dr. Xiaozhou Li (HP Labs) for agreeing to take time out of his busy schedule to be part of my dissertation committee, despite being in a different time zone. I also thank him for his careful advice, encouragement, and his support. I thank Dr. Michael J. Carter (UNH ECE) for agreeing to be a member of my dissertation committee. I appreciated his timely advice and encouraging words throughout the dissertation process. His careful feedback on the text of my dissertation helped in improving it significantly.

I thank Prof. Michel Charpentier for his collaboration and guidance, and for showing me how to think and write clearly. I aspire to his level of clarity and organization. I thank him for always taking my vague ideas seriously and according me the privilege of discussing, thinking, and writing with him. I thank Prof. Elizabeth Varki for being a constant source

of support. I thank her for teaching me about storage and operating systems and some queuing theory. I am in awe of her ability to present complex ideas in clear but compelling language. I aspire to her courage and her confidence in taking on tremendous challenges.

I am honored to have had an opportunity to study with Prof. Pilar de la Torre. She taught me all the theory and mathematics that I know and how to recognize what I don't know and learn it. Without her inspiring and encouraging words, her incredibly insightful advice and foresight, and her patient and caring guidance, I could not have come this far. I thank her for the wonderful debt that I owe her.

I thank Prof. Radim Bartoš, my advisor, and friend, since my early days at UNH, and one of the best advisors at UNH. I learned about networks, research, writing, presenting, grant writing, teaching, and communication from him without ever hearing a discouraging word from him. He understood the conditions that would allow me to perform my best and worked really hard to provide them. His office door was always open to me and I felt that there was nothing that I could not discuss with him. Every meeting left me with a new perspective and renewed my enthusiasm and confidence. I wish to thank him for accepting me as his student and patiently showing me the way through, what became some of the best years of my life. Here, I do not wish to offer the worn truth that one's debt to one's teachers is irrepayable, but rather hope only that my little contributions kept my teachers, collaborators, and friends reasonably amused and intrigued.

I thank Prof. Philip J. Hatcher for the countless times that he bailed me out of difficult situations, and for his advice and guidance throughout the Ph.D. process. I also thank him for giving me the opportunity to teach at UNH, an experience that I shall always cherish. (I also wish to thank my fantastic students for adapting to a novice teacher without complaint.) I thank Prof. Daniel Bergeron for agreeing to serve on my depth examination committee and Prof. Colin Ware for his course in visualization and for our subsequent short but extremely fun collaboration. I thought I knew programming, until I took Prof. Robert Russell's operating system programming course—he taught me how to program. Prof. Charpentier's course changed the way I think about programs. Prof. Ernst Linder

(Dept. of Mathematics and Statistics) taught me all the statistics that I know. I am grateful for having such excellent teachers and I highly recommend their courses.

I thank the Graduate School for their support throughout my studies and Dean Moorhead for her faith in me. I thank Leila Paje-Manalo and Cathryn Spreeman of the Office of International Students and Scholars for their vigilant help in ensuring that my stay was always welcome. I thank Linda Spring-Andrews, Rachel Purnell, and Carolyn Kirkpatrick for doing a fantastic job running the department office. I also thank UNH Housing for supporting Babcock House—especially Paula Dinardo and Victoria Wilson for making it a diverse and fun community. Because I was fortunate to have these fantastic people looking after me, I enjoyed my stay at UNH, in New Hampshire, and in the United States. Thank you for making me feel welcome.

Lastly, I wish to thank those to whom this work is dedicated: my teachers for their guidance and encouragement; my friends for their steadfast friendship; Katie, for her unwavering companionship and affection; and my family, for their faith and patience, while I chased my dream.

Table of Contents

List of Tables

List of Figures

ABSTRACT

TOPICS IN ACCESS, STORAGE, AND SENSOR NETWORKS

by

Swapnil Bhatia
University of New Hampshire, May, 2010

In the first part of this dissertation, Data Over Cable Service Interface Specification (DOCSIS) and IEEE 802.3ah Ethernet Passive Optical Network (EPON), two access networking standards, are studied. We study the impact of two parameters of the DOCSIS protocol and derive the probability of message collision in the 802.3ah device discovery scheme. We survey existing bandwidth allocation schemes for EPONs, derive the average grant size in one such scheme, and study the performance of the shortest-job-first heuristic.

In the second part of this dissertation, we study networks of mobile sensors. We make progress towards an architecture for disconnected collections of mobile sensors. We propose a new design abstraction called *tours* which facilitates the combination of mobility and communication into a single design primitive and enables the system of sensors to reorganize into desirable topologies after failures. We also initiate a study of computation in mobile sensor networks. We study the relationship between two distributed computational models of mobile sensor networks: population protocols and self-similar functions. We define the notion of a self-similar predicate and show when it is computable by a population protocol.

Transition graphs of population protocols lead us to the consideration of graph powers. We consider the direct product of graphs and its new variant which we call the lexicographic direct product (or the clique product). We show that invariants concerning transposable walks in direct graph powers and transposable independent sets in graph families generated by the lexicographic direct product are uncomputable.

The last part of this dissertation makes contributions to the area of storage systems. We propose a sequential access detection and prefetching scheme and a dynamic cache sizing

scheme for large storage systems. We evaluate the cache sizing scheme theoretically and through simulations. We compute the expected hit ratio of our and competing schemes and bound the expected size of our dynamic cache sufficient to obtain an optimal hit ratio. We also develop a stand-alone simulator for studying our proposed scheme and integrate it with an empirically validated disk simulator.

Part I

Protocols in access networks

Chapter 1

Introduction to Access Networks

An *Access Network* can be defined as a network of resources, shared by a geographically clustered group of a large number of non-cooperating users, that makes locally accessible to these users, the services of a global network. Public transport networks such as bus and train networks which connect residents of neighboring towns to an international airport are general examples of an access network. In the realm of data transport networks, Digital Subscriber Line (DSL), DOCSIS (cable-modem), and PSTN-based dial-up networks are some particular examples of prevalent access networks. The focus of this work is the study and design of data access networks (henceforth, referred to merely as access networks). Over the years, the inadequacy of current access networks in its many dimensions has become increasingly evident. Some of the main challenges are: 1) *Scarcity of the available data bandwidth:* Dial-up, DSL, DOCSIS as well as most of the current wireless technologies do not offer sufficiently high bandwidth to support applications of the future. 2) *Guaranteed levels of service:* Although this problem has remained at the center of Internet research for a decade, very little (if any) of the knowledge gained has been implemented into access networks. As a result, current networks are still incapable of providing service guarantees for diverse applications. 3) *Efficient and optimal control:* Design of optimal bandwidth control and allocation algorithms for access networks as defined above is a difficult problem. The difficulty arises partly due to the high bandwidth and the large distances covered by access networks. The goal of a high-performance public access

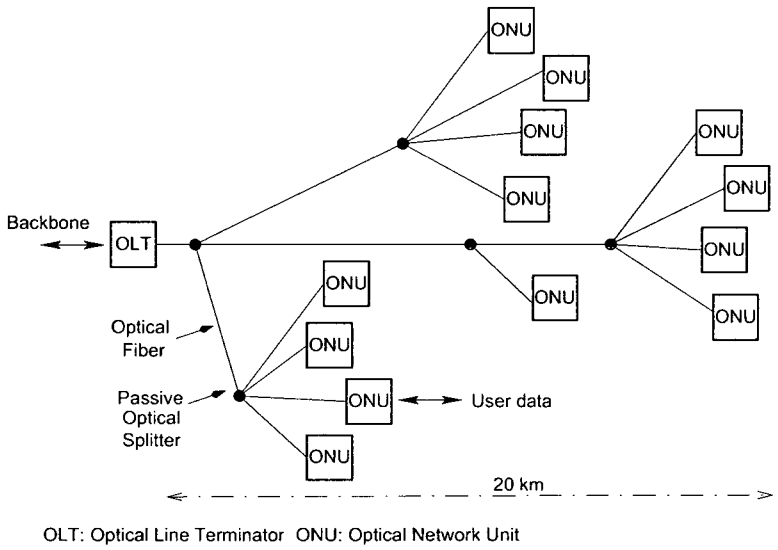


Figure 1-1: An Ethernet Passive Optical Network

network has not been achieved largely due to the two above challenges. 4) *Near-permanent scalability:* This is the most important challenge from the perspective of network operators who must incur the high initial cost of deploying a new access network. Current access networks are tied to the "legacy" physical infrastructure available from CATV and PSTN networks. Consequently, these access networks have a short life-span and new networks will have to be built in the very near future. This is evident from the recent ubiquity of broadband optical and wireless access in many developed countries in Asia and Europe. 5) *Reliability:* By virtue of being a public service infrastructure, an access network must be reliable. Current access networks score low as far as service downtime is concerned. 6) *Security:* Finally, almost all of the current access networks have security added as an afterthought to the initial design. Hence, *inherent* security in access networks is the final challenge.

1.1 Ethernet Passive Optical Networks (EPONs)

An *Ethernet Passive Optical Network (EPON)* is a point-to-multipoint, bidirectional, high rate, long-range optical network for data communication. It is one of the new network designs proposed to address some of the challenges faced by existing access network designs. An EPON link is shared by multiple users. Each user connects to the EPON link through a device known as an *Optical Network Unit (ONU)*. Since the link is shared, its use must be coordinated. This function is performed centrally by a single special device called the *Optical Line Terminator (OLT)*. The direction of communication from the ONUs to the OLT is known as the *upstream* direction and the opposite direction is known as the *downstream* direction. The link is organized as a tree topology at the physical layer with the OLT situated at the root of the tree, the ONUs attached as its leaves, and optical splitters forming the interior nodes of the tree. For definitions of EPON terms used in this proposal, we refer to the IEEE 802.3ah standard [3].

1.1.1 ONU Discovery

Before an ONU is able to use the EPON link for data transmission, it must register itself with the OLT over the EPON link without human intervention. Since this is the first communication between the ONU and the OLT, the exact latency in propagating a message in the upstream or the downstream direction is known neither to the OLT nor the ONU. An ONU may be located at an arbitrary (but bounded) distance from the OLT. The exact latency in propagating a message between them may vary over time due to physical changes in the transmission medium. Consequently, messages from two ONUs located at similar distances from the OLT may overlap and their reliable recovery at the OLT cannot be guaranteed. For registration to occur without human intervention, the IEEE 802.3ah protocol suite includes a randomized discovery

phase, the primary goal of which is to enable the OLT to gather timing information for each ONU. In this phase, the OLT broadcasts to all ONUs, the length of an interval of time reserved for transmission of registration requests by new ONUs. Such ONUs, upon receiving this message, wait for a random interval of time (chosen uniformly at random within the stipulated length), and then transmit their registration request message. (ONUs registered already must remain silent and queue packets during the discovery phase.) If the registration message transmitted by a new ONU is correctly received by the OLT, then the registration process continues and the ONU becomes registered with the OLT. As a side effect of successful registration, the OLT deduces the round-trip latency to the source ONU and uses it to communicate with the ONU individually from then on. If two different ONUs choose to transmit their registration requests at roughly the same time, then the messages arrive overlapped and illegible at the OLT. In this case, both ONUs retry registration in the next discovery phase.

1.1.2 Analysis of the randomized discovery scheme

The performance of the randomized discovery scheme described above has not been investigated by others in the available literature. Previous work on the performance analysis of random access protocols [112, 21] makes assumptions about the arrival process that are unsuitable in the context of the EPON discovery scheme. Previous analyses of similar random access schemes involve a single random variable representing the time of message transmission by a device. The EPON discovery scheme involves two independent random variables: the time of transmission and the propagation delay before the message reaches the point of contention at the OLT receiver. The EPON discovery scheme is not specific to EPONs; similar discovery schemes are required in other access network technologies such as Data-over-cable (DOCSIS), and WiMax (IEEE 802.16).

Specifically, the following questions must be addressed:

1. How effective is the randomized discovery phase? What is the expected number of ONUs that are discovered in an instance of a discovery phase?

2. How does the discovery scheme perform in the presence of a large number of ONUs? Is there an optimal length of the discovery interval?

These questions are answered in Chapter 2.

1.1.3 Online Bandwidth allocation in EPONs

After an ONU has registered, it joins the pool of ONUs sharing the EPON link. In the downstream direction, the total bandwidth of the link is shared by all users and the transmission of their data is performed singly by the OLT. In the upstream direction, the bandwidth is also shared and the transmission of the data for each user is performed individually by each ONU. The OLT decides which ONU is allowed to transmit data, the number of bytes that it is allowed to transmit, and the time at which it should begin its transmission. The OLT uses a special control message called a *Gate* to grant transmission opportunities to ONUs. With the data traffic, the ONU also transmits a control message containing a *Report* of the number of bytes buffered in its queue waiting for a subsequent transmission opportunity. An algorithm implemented in the OLT, which uses these reports and gate messages to construct an upstream transmission schedule is known as a dynamic bandwidth allocation (DBA) algorithm. (In this proposal, our primary interest is in the design of the upstream DBA; hence we will not discuss the downstream DBA.)

A simple DBA algorithm is the following: the OLT grants an immediate transmission opportunity to an ONU by sending it a gate message. The ONU, upon receiving the Gate message immediately begins transmitting data, continues for the allotted

duration, and before the end of the opportunity sends a report of the number of bytes waiting in its queue to the OLT. After the current ONU has finished its transmission, the OLT records its report and then grants a transmission opportunity to a different ONU. The same sequence of events as above repeats and this continues until all ONUs have been served a transmission opportunity. The OLT then starts a new round of grants.

An immediately apparent disadvantage of this simple scheme is the following: during the time interval between the OLT's sending the gate message and the ONU receiving it, and the time interval between the ONU sending the first bit of its data and the OLT receiving it, the upstream portion of the EPON link remains unutilized. To ensure high upstream link utilization, a DBA algorithm must interleave the transmission of a gate message to an ONU with the reception of data from a previous ONU. Such a scheme was first proposed by Kramer et al. [71, 72]. In this scheme called IPACT, ONUs are served in a round-robin manner with downstream grant messages for future transmissions temporally interleaved with current upstream data transmissions. The length of a transmission opportunity allocated to an ONU is decided by an allocation policy. IPACT with a static allocation policy grants a fixed length transmission opportunity to each ONU regardless of the number of bytes reported waiting in their buffer. A gated allocation policy grants to the ONU a transmission opportunity exactly equal to the number of waiting bytes last reported by the ONU. Limited allocation is an upper bounded variant of the the gated allocation policy. A credit-based allocation policy may grant a transmission opportunity sufficient to transmit a larger number of bytes than that last reported by the ONU, thus compensating for the staleness of the information contained in the report. Since IPACT, many DBA schemes have been proposed. We provide an overview of these schemes in Chapter 3. Key performance measures that these schemes have sought to

improve are utilization, delay, and fairness, and they have done so through improved gate transmission scheduling; traffic prediction, classification and prioritization; and excess bandwidth redistribution. In Chapter 5, we propose a new DBA scheme based on the shortest-job-first scheduling heuristic.

1.1.4 An analytical model of IPACT

Although IPACT was the first DBA scheme proposed for EPONs, it has been studied only through simulations. There has been no analytical model of the bandwidth granting mechanism in IPACT under the gated allocation policy with symmetric traffic. Investigating DBA schemes analytically is complicated by the dependence of the length of the transmission grant allocated to an ONU on the grants allocated to it and other ONUs in the past. The goal of this research work is to capture the behavior of IPACT using a simple set of equations relating the traffic load to the size of the grant allocated by IPACT. We present our results in Chapter 4.

1.2 CATV-based access networks

The earliest broadband access networks used existing cable television wiring as the physical transmission medium. These networks are pervasive today and subscribers know them as cable modem-based networks. A majority of these networks are based on the Data Over Cable Service Interface Specification (DOCSIS) developed by CableLabs in the late 90s. The structure of a DOCSIS network is similar to that of an EPON: a head-end known as a Cable Modem Termination System (CMTS) resides at the central office and manages one or more physical interfaces to a coaxial cable network. The cable carries legacy television channels and DOCSIS channels. Subscribers connect to the network through a Cable Modem (CM) which acts as a router or a bridge between the subscriber's network and the service provider's DOCSIS net-

work. Each CMTS is responsible for a single downstream channel (from the CMTS to the CM) and several upstream channels. Each upstream channel may be shared by several CMs and each CM may use a single upstream channel at any time. Thus upstream bandwidth on each channel must be arbitrated by the CMTS. The DOCSIS standard provides a suite of protocols for registration and bandwidth management. As per the protocol, the CMTS can issue several different types of grants to a CM depending on its type of traffic. Orthogonally, the CMTS can also turn on and off certain features associated with these grant types. This multitude of choices makes DOCSIS a complicated protocol and makes it difficult to choose a set of features appropriate for a particular deployment. In Chapter 6, we experimentally evaluate the performance of two such features: piggybacking and concatenation. We exhaustively evaluate the appropriateness of each combination of these features and show that concatenation is beneficial only for short packets and that piggybacking can be detrimental to throughput unless the number of CMs is sufficiently large.

Chapter 2

Closed-Form Expression for the Collision Probability in the IEEE EPON Registration Scheme

We derive a closed-form expression for the message collision probability in the IEEE 802.3ah Ethernet Passive Optical Network (EPON) registration scheme. The expression obtained, although based on an approximation, shows a good match with simulation results. We use the results of our analysis to compute the size of the most efficient contention window and the most efficient number of nodes serviced by a given window size.

2.1 Motivation

Protocols for emerging access network technologies such as DOCSIS [1], EPON [3] and some wireless technologies include a preliminary phase where the subscriber device must register with a headend or base station residing at a central office. Since this is the first communication between the headend and the subscriber device, no information about key parameters such as latency or timing is available to either party. Subscriber devices may be located at random distances unknown to the headend. As a result, most protocols rely on some collision avoidance scheme in order to reduce

contention in the use of the communication channel. The recently adopted IEEE 802.3ah EPON standard prescribes the Random Delay scheme for this purpose. In this scheme, the headend broadcasts the size of a contention interval. The nodes, upon receiving this message, wait for a uniformly random interval and then transmit their registration message. In this work, we derive a closed-form expression for the probability of message collision in this scheme and validate our result through simulation.

To our knowledge, this is the first attempt at computing the probability of collision for the IEEE EPON registration scheme. While previous work in this area [112, 21] serves as an excellent general reference, its focus has primarily been on the stability and throughput of multiaccess schemes. Moreover, most of the assumptions (Poisson arrivals, backlogged nodes, etc.) are either not relevant to the IEEE EPON registration scheme or are out of scope of our current work. For example, before the average success probability in a single registration cycle—the focus of our present work—is known, the multi-step performance of the scheme cannot be analyzed. Our own past work [22] focuses on the high load performance characterization of the IEEE EPON registration scheme through simulations. A more recent analysis [68] is restricted only to the simpler case of identically distanced nodes. We propose a more generic model applicable to identically distanced as well as randomly distributed nodes. Our model includes the random round trip delay together with the random contention window size. The model is parameterized by message size, contention window size and round trip time and is therefore directly applicable to a practical analysis of the IEEE EPON registration protocol.

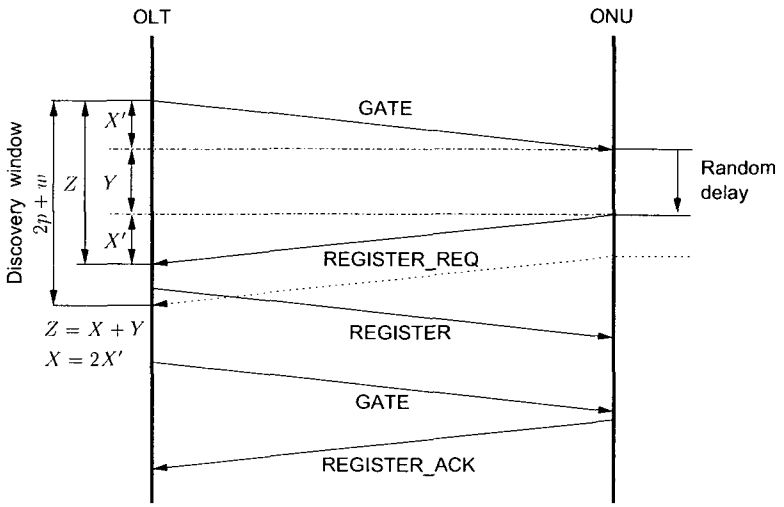


Figure 2-1: The IEEE EPON Registration scheme.

2.2 Modeling Registration of an IEEE EPON Device

Fig. 2-1 illustrates the IEEE EPON registration scheme. In this work, we will focus only on the first step in the registration scheme: the transmission of the registration request. Subsequent steps cannot occur until the first step is completed successfully. As per the EPON protocol, the headend broadcasts a discovery message to signal the beginning of a special interval reserved for new-node registration. A new node, upon receiving it, replies with a registration request message transmitted after a random wait. If two such registration messages, say of length k each, arrive at the headend overlapping in time, then there is a collision. Thus, to model a collision we must calculate the arrival time of a message at the headend. We observe that this arrival time is a sum of the one-way propagation time of the broadcast message from the headend to the node, a random wait at the node, and another one-way propagation time of the registration request message from the node to the headend.

Due to technological constraints on the power and reach of a transmitted signal, the IEEE EPON standard [3] fixes the maximum distance from the headend at which a node may be located. We assume the maximum reach of our network to be such as

to result in a maximum one-way propagation time of p and the maximum random wait time to be w (also fixed by the headend). Thus, the arrival time can vary between a minimum of 0 and a maximum of $2p + w$ as shown in Fig. 2-1. However, since the IEEE EPON standard [3] places no constraints on the contention window size $w \geq 0$, we let $M = \max(2p, w)$ and $m = \min(2p, w)$. Thus, $M \geq m$.

Let X, Y and Z be random variables. Let X, Y be independent and have a uniform distribution with $X \in \text{Uniform}[0, M]$, $Y \in \text{Uniform}[0, m]$, $M \geq m \geq 0$. Depending on the magnitudes of p and w, X and Y will interchangeably model the two-way propagation time and the random wait. To simplify the exposition, we first consider $m > 0$ and add $m = 0$ as a separate case later. Let f_X, f_Y denote their probability mass function of X and Y respectively. Thus, $f_X(x) = M^{-1}$ and $f_Y(y) = m^{-1}$. Let $Z = X + Y$. Thus, Z models the arrival time of a message from a node at the headend as shown in Fig. 2-1. Since X and Y are independent, their joint density $f_{XY}(x, y) = (Mm)^{-1}$. Let $F_Z(z) = P(Z \leq z)$ denote the cumulative distribution function (CDF) of Z. We compute the CDF of Z, by integrating $P(X+Y = Z)$ w.r.t z [89]. Our main derivation involves the computation of many such integrals and we omit the details of these calculations due to space constraints. The limits used for each integral are specified in the accompanying figures and should aid the reader in computing the integrals, if desired. Thus, we have:

$$F_Z(z) = \begin{cases} 0 & \text{if } z < 0 \\ z^2/2Mm & \text{if } z \leq m \\ (2z - m)/2M & \text{if } m < z \leq M \\ (2zM + 2zm - m^2 - z^2 - M^2)/2mM & \text{if } M < z \leq M + m \\ 1 & \text{if } z > M + m. \end{cases} \tag{2.1}$$

To find the probability mass function $f_Z(z)$, we differentiate w.r.t. z to get:

$$f_Z(z) = \begin{cases} 0 & \text{if } z < 0 \\ f_1(z) = z/mM & \text{if } z \leq m \\ f_2(z) = 1/M & \text{if } m < z \leq M \\ f_3(z) = (m + M - z)/mM & \text{if } M < z \leq M + m \\ 0 & \text{if } z > M + m. \end{cases} \tag{2.2}$$

2.3 Formulating the Collision Event

Let n denote the total number of devices attempting to send their respective registration request messages. Let Z_i denote the arrival time of the message from the device $1 \leq i \leq n$ at the headend. (We will use Z_i to refer to the device i as well as the random variable, depending on the context.) Since all the devices behave identically, any result for a single device will be true for any device. Fix Z_1 as the device under observation. Then, a successful transmission by Z_1 of length k can be expressed as the event:

$$\begin{aligned} \bigcap_{i=2}^{n} \big[|Z_1 - Z_i| > k\big] &= \bigcap_{i=2}^{n} \big[(Z_1 - Z_i > k) \cup (Z_1 - Z_i < -k)\big] \\ &= \bigcap_{i=2}^{n} \big[(Z_i < Z_1 - k) \cup (Z_i > Z_1 + k)\big]. \end{aligned} \tag{2.3}$$

Suppose $Z_1 = t$ where $0 \leq t \leq M + m$, i.e., the transmission from device 1 arrives at the headend at some time, t. Under this condition, a successful transmission event for Z_1 can be expressed as:

$$\bigcap_{i=2}^{n} \big[|Z_1 - Z_i| > k | Z_1 = t\big] = \bigcap_{i=2}^{n} \big[(Z_i < t - k) \cup (Z_i > t + k)\big]. \tag{2.4}$$

Since all the devices follow the same registration protocol and transmit registration messages independently of each other, the Z_i are all i.i.d. with densities described by Eqn. (2.1). Therefore, the probability of a successful transmission by Z_1 can be

expressed as:

$$P\Big(\bigcap_{i=2}^{n}\big[|Z_1 - Z_i| > k | Z_1 = t\big]\Big) = \prod_{i=2}^{n} P\big[(Z_i < t-k) \cup (Z_i > t+k)\big] = P\big[(Z_2 < t-k) \cup (Z_2 > t+k)\big]^{n-1}, \quad (2.5)$$

where we use Z_2 to represent any single other device. Since

$$P[(t-k) < Z_2 < (t+k)] = F_{Z_2}(t+k) - F_{Z_2}(t-k),$$

the remaining probability in the "tails" can be expressed as:

$$P\big[(Z_2 < t-k) \cup (Z_2 > t+k)\big]^{(n-1)} = \big[1 - [F_{Z_2}(t+k) - F_{Z_2}(t-k)]\big]^{(n-1)}. \quad (2.6)$$

Hence, probability of a successful transmission by Z_1 given that $Z_1 = t$ is

$$P\Big(\bigcap_{i=2}^{n}\big[|Z_1 - Z_i| > k | Z_1 = t\big]\Big) = \big[1 - [F_{Z_2}(t+k) - F_{Z_2}(t-k)]\big]^{(n-1)}. \quad (2.7)$$

Finally, applying the Law of Total Probability,

$$P\Big(\bigcap_{i=2}^{n}\big[|Z_1 - Z_i| > k\big]\Big) = \int_{-\infty}^{\infty} P\Big(\bigcap_{i=2}^{n}\big[|Z_1 - Z_i| > k | Z_1 = t\big]\Big) \cdot P(Z_1 = t) \cdot dt = \int_{-\infty}^{\infty} [1 - F_{Z_2}(t+k) + F_{Z_2}(t-k)]^{(n-1)} \cdot f_{Z_1}(t) \cdot dt. \quad (2.8)$$

$F_{Z_2}(z)$ and $f_{Z_1}(z)$ are available from Eqns. 2.1 and 2.2. Fig. 2-2 shows the probability mass function $f_{Z_i}(z)$ of any device Z_i along with an illustration of the condition $Z_1 = t$. From Fig. 2-2, we can see that due to the piecewise structure of $f_{Z_i}(z)$, the limits for integrating w.r.t $Z = t$ will depend heavily on the relative magnitudes of k, m and M. Moreover, the limits will also depend on the position of t relative to k, m and M. For example, consider the simplest case where $m = 0, M > 0$ and $0 < k \leq M - k$. For this one permutation of parameters, we must again split the calculation of Eqn. (2.8) for

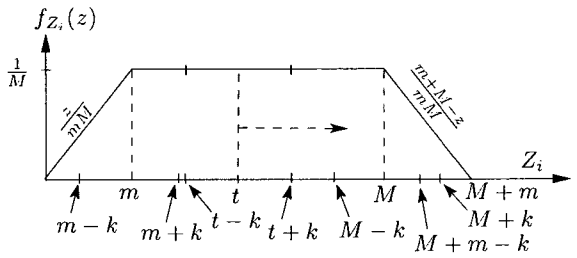


Figure 2-2: Probability mass function $f_{Z_i}(z)$ with the condition $Z_1 = t$.

various relative positions of t w.r.t the remaining intervals shown in Fig. 2-2. Thus, we would have to consider separately $t < 0, 0 \leq t < k, k \leq t < M-k, M-k \leq t < M$ and $M \leq t < M+k$. Note that these subintervals are specific to the single simple case above. For the other more complex cases, the description of subintervals to be considered will be different and their number will be larger.

Clearly, the present approach will lead to a large number of subintervals over which the calculation of Eqn. (2.8) will have to be performed—a tedious process. Notice that out of all the cases, those introduced due to the relative magnitudes of our parameters k, m and M are unavoidable. However, the subcases due to the conditional $Z_1 = t$ have been introduced only because the events in Eqn. (2.3) are not independent. If an assumption were made as to the independence of these events, then a number of subcases would be avoided. Specifically, all the subintervals introduced due to the conditional $Z_1 = t$ could be avoided at the expense of introducing some error into the calculation. In the next section, we explore this approach and obtain an approximation which shows a good match with values obtained from simulations.

2.4 Derivation of an Approximating Expression

Consider again, two i.i.d. random variables Z_1 and Z_2 with probability mass functions $f_{Z_1}(z_1)$ and $f_{Z_2}(z_2)$ as derived in Eqn. (2.2). Consider their joint density. Due to the piecewise structure of $f_Z(z)$, the joint density of Z_1 and Z_2 will comprise of nine regions defined by the three cases each for Z_1 and Z_2 as illustrated in Fig. 2-3. Fig. 2-3 (right) shows the joint density of Z_1 and Z_2 for examples $M = 10\ \mu s$ and $m = 5\ \mu s$. As is clear from Eqn. (2.2), the joint density will plateau for $m \leq Z_1, Z_2 \leq M$ when $m < M$. The shaded area in Fig. 2-3 (left) shows the region where $|Z_1 - Z_2| \leq k$. Let P be the event $|Z_1 - Z_2| \leq k$. To find the probability of the event P, as mentioned

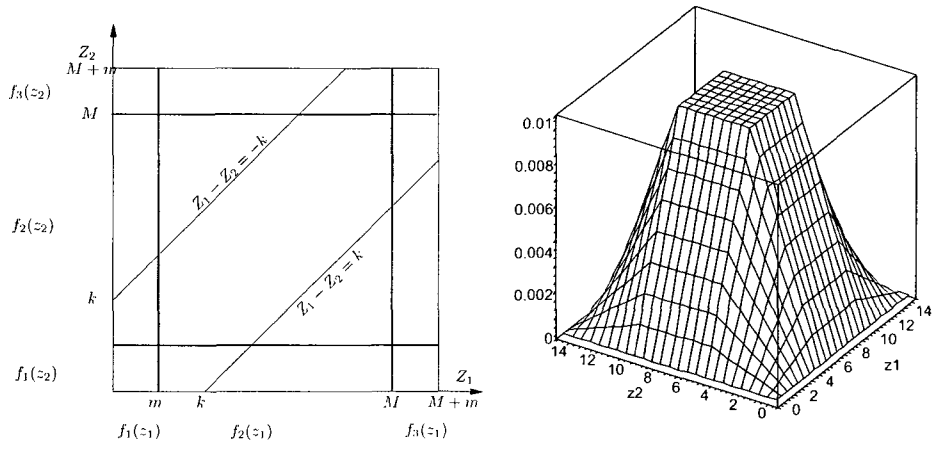


Figure 2-3: *Left*: Computing the probability of $|Z_1 - Z_2| \leq k$. *Right*: Joint density of Z_1, Z_2 (in μs) with $M = 10, m = 5$.

earlier, we must consider several cases arising from the magnitudes of the parameters k, m and M relative to each other. In each case, we also need to consider whether $m, M \leq k$ or $m, M > k$. Finally, $M + m < k$ or $m = 0$ are other special cases. Together, all the scenarios result in a function of the form expressed in Fig. 2-4. Due

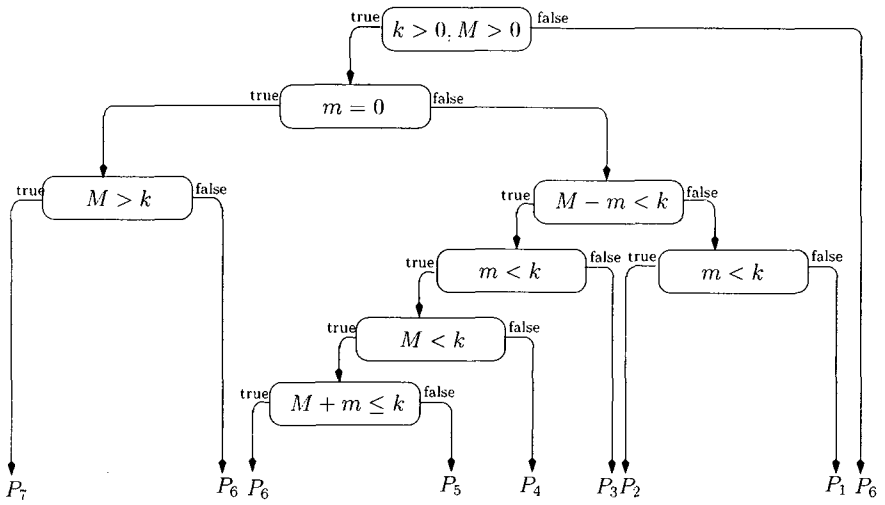


Figure 2-4: $P(|Z_1 - Z_2| \leq k)$.

to the absence of the conditional, the number of cases to be considered is significantly reduced. Note that were the conditional present, each branch of the tree in Fig. 2-4 would result in many more branches. Due to our assumption about the independence of events in Eqn. (2.3), we are able to prune the tree much earlier.

Figs. 2-5—2-11 illustrate each of the P_i in Fig. 2-4. Using these figures, we calculate the probability contained in the shaded region for each P_i. Integrating piecewise within the limits assigned to the shaded region in each figure, we obtain the following expressions for each of the P_i in Fig. 2-4:

$$P_1 = \frac{k(k^3 - 4m^3 - 4mk^2 + 12Mm^2)}{6m^2M^2} \tag{2.9}$$

$$P_2 = \frac{(12Mk - m^2 - 6k^2)}{6M^2} \tag{2.10}$$

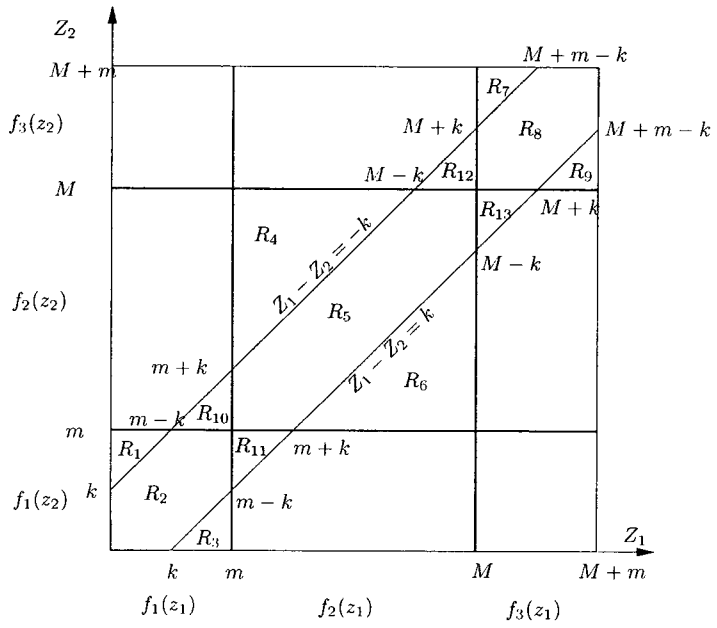


Figure 2-5: P_1: $m \geq k, M > m$ and $M - m \geq k$.

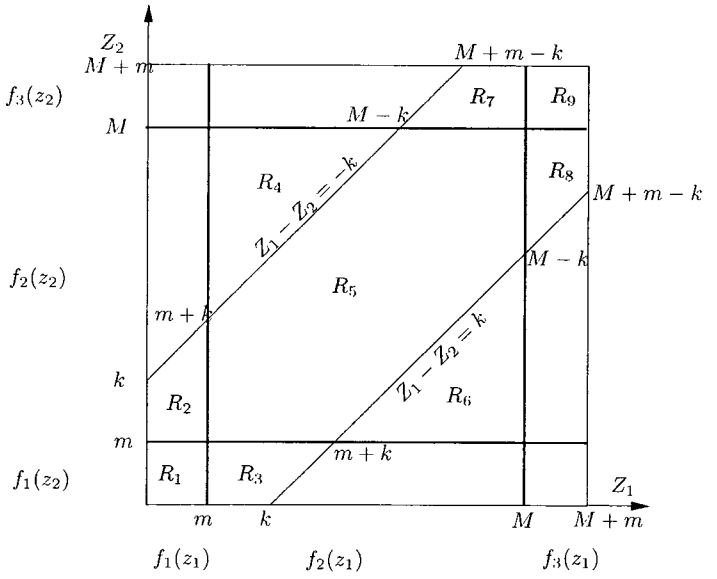


Figure 2-6: P_2: $m < k, M > m$ and $M - m \geq k$.

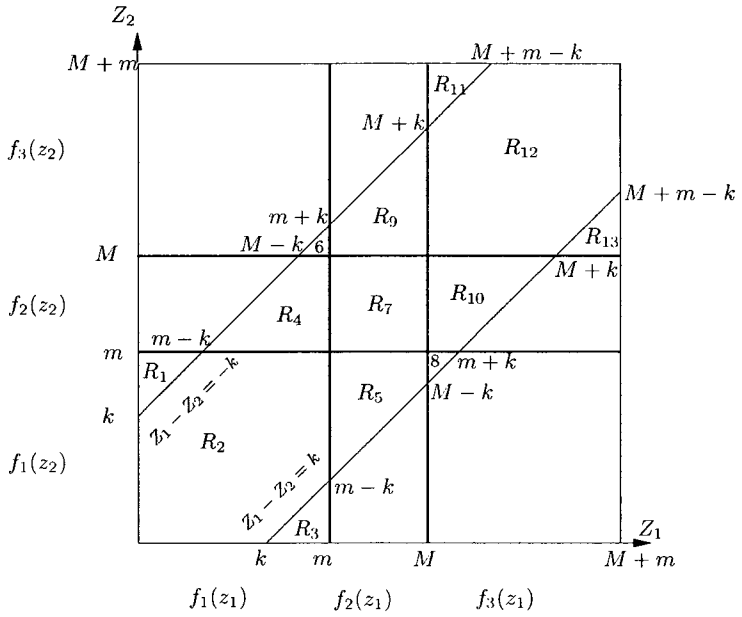


Figure 2-7: P_3: $m \geq k, M \geq m$ and $M - m < k$.

$$P_3 = \left(\frac{12Mkm^2 + 12mkM^2 - 12mk^2M + 6m^2M^2 + m^4 + 3k^4 - 4km^3}{12m^2M^2} + \frac{6k^2m^2 - 4mk^3 - 4kM^3 + 6k^2M^2 - 4Mk^3 + M^4}{12m^2M^2} - \frac{4mM^3 + 4Mm^3}{12m^2M^2} \right) \tag{2.11}$$

$$P_4 = \left(\frac{6m^2M^2 - 4mM^3 + 12mM^2k - 12mMk^2 + 4mk^3 - m^4}{12m^2M^2} + \frac{12Mkm^2 - 4M^3k + 6M^2k^2 - 4Mk^3 + 4km^3 - 4Mm^3}{12m^2M^2} + \frac{k^4 + M^4 - 6k^2m^2}{12m^2M^2} \right) \tag{2.12}$$

$$P_5 = \left(\frac{12km^2M + 12kmM^2 - 12k^2mM + 6m^2M^2 - k^4 - 4m^3M - 4mM^3}{12m^2M^2} + \frac{4mk^3 + 4Mk^3 - M^4 - m^4 + 4km^3 + 4kM^3 - 6k^2m^2 - 6k^2M^2}{12m^2M^2} \right) \tag{2.13}$$

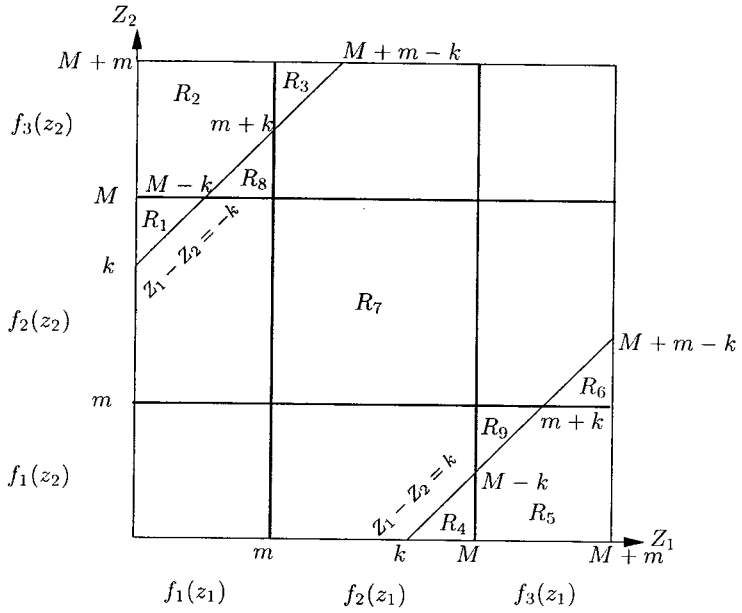


Figure 2-8: P_4: $m < k, M > m, M \geq k$ and $M - m < k$.

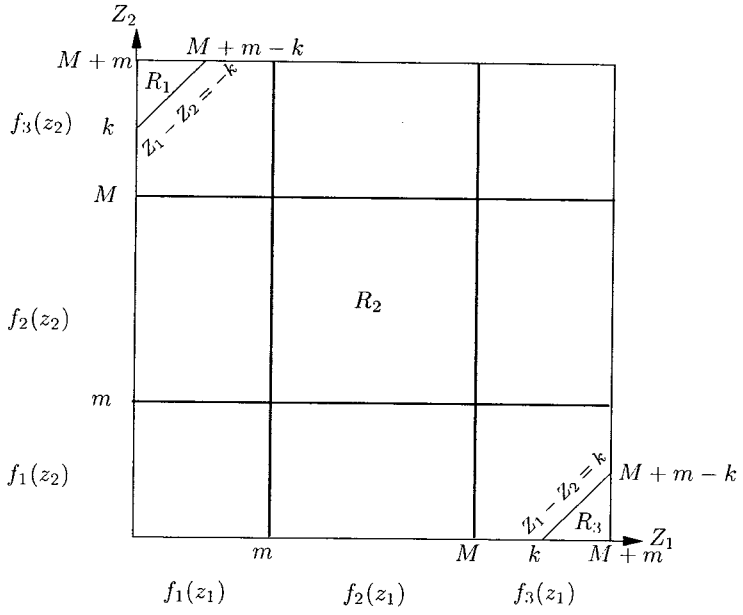


Figure 2-9: P_5: $m < k, M \geq m, M < k$ and $M - m < k, M + m > k$.

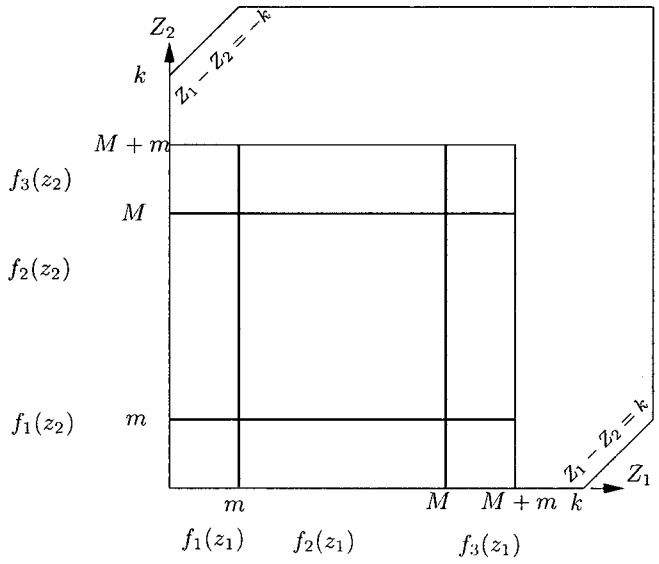


Figure 2-10: P_6: $m < k, M \geq m, M < k$ and $M - m < k, M + m \leq k$.

$$P_6 = 1 \tag{2.14}$$

If $m = 0$, then $Y \in \text{Uniform}[0, 0]$ and hence $Z = X$. Therefore, $f_Z = f_X$, and:

$$P_7 = \frac{k(2M - k)}{M^2}. \tag{2.15}$$

Fig. 2-12 shows the probability of collision for two nodes participating in the IEEE EPON registration scheme with a total message length of 316 bytes (64-byte actual message with additional overhead [68]) as specified in the IEEE EPON standard (equivalent to $k = 2.528$ μs). While the value for the parameter p is also specified in the standard as 100 μs (equivalent to 20 km), the range of values for p in the figure allows us to use the same model to compute the probabilities for clustered nodes or nodes situated at an identical distance. The range of values for the wait period w is unspecified by the standard and is open to various implementation schemes.

We now extend our 2-node model to n nodes. Let $P_s(k)$ and $P_c(k)$ denote the probability of successful and unsuccessful transmission (i.e. collision) respectively for

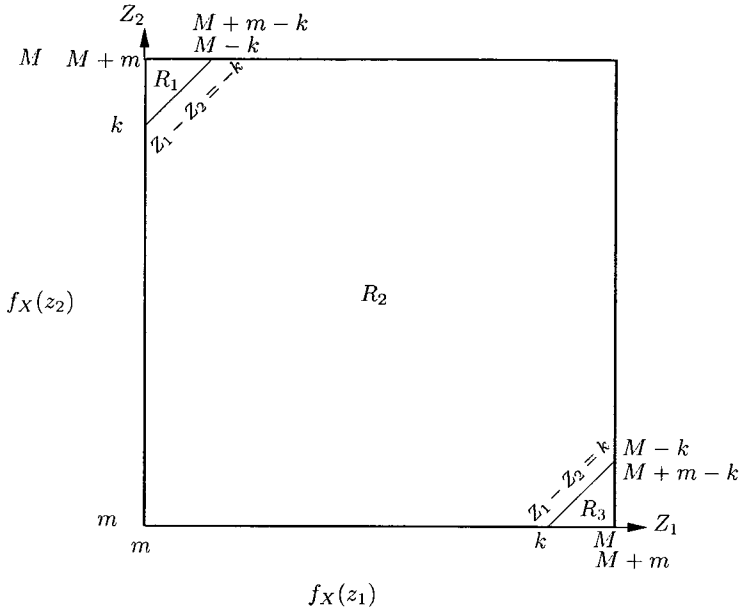


Figure 2-11: P_7: $m = 0, M > k$.

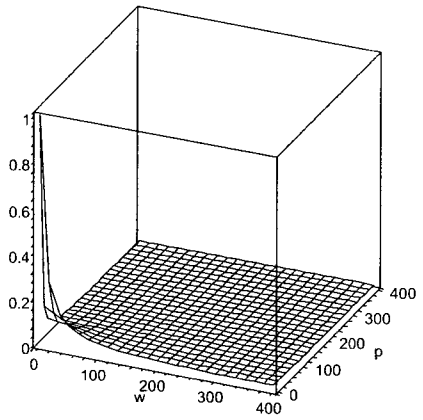


Figure 2-12: Probability of collision with two randomly distanced nodes (p, w in μs).

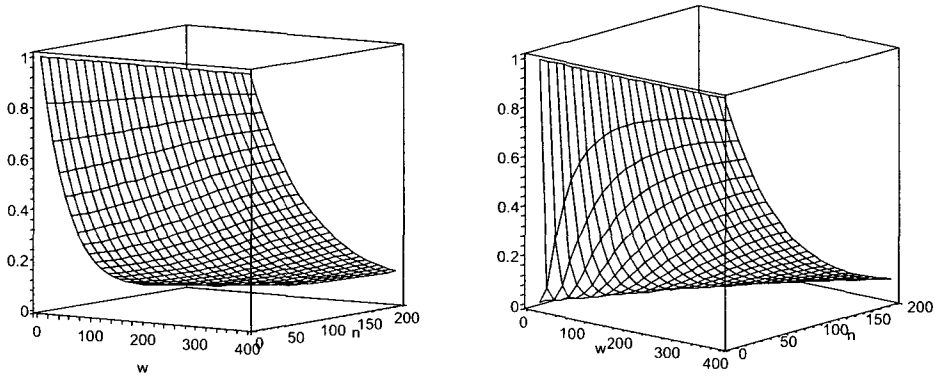


Figure 2-13: Probability of successful transmission with n randomly distanced *(left)* and n identically distanced *(right)* nodes. (w is in μs.)

a node in presence of $k-1$ other nodes. The probability of successful transmission in the 2-node case is thus $P_s(2) = 1 - P_c(2)$. In Sec. (2.4), we derived $P_c(2)$ since $P_c(2) = P(|Z_1 - Z_2| \leq k)$. A successful transmission by a node in the presence of $n-1$ other nodes implies that its transmission did not collide with any of the other $n-1$ nodes. If we assume independence of each pairwise collision event of Eqn. (2.8), we can write:

$$P_s(n) = P_s(2)^{n-1}. \tag{2.16}$$

This key assumption allows us to sidestep calculation of Eqn. (2.8) over the many subintervals. Our calculations are now restricted only to the different cases introduced by the parameters k, m and M. Fig. 2-13 (left) shows the probability of successful transmissions for 1 to 200 nodes for a range of waiting times. The propagation time p is set to 100 μs (20 km). We can also formulate the scenario where all the nodes are at an identical distance by setting $p = 0$. Fig. 2-13 (right) shows the performance of the scheme for 1 to 200 nodes located at identical distances. Figs. 2-14 and 2-15 compare the results from simulation plotted with those from our closed-form expression. Our

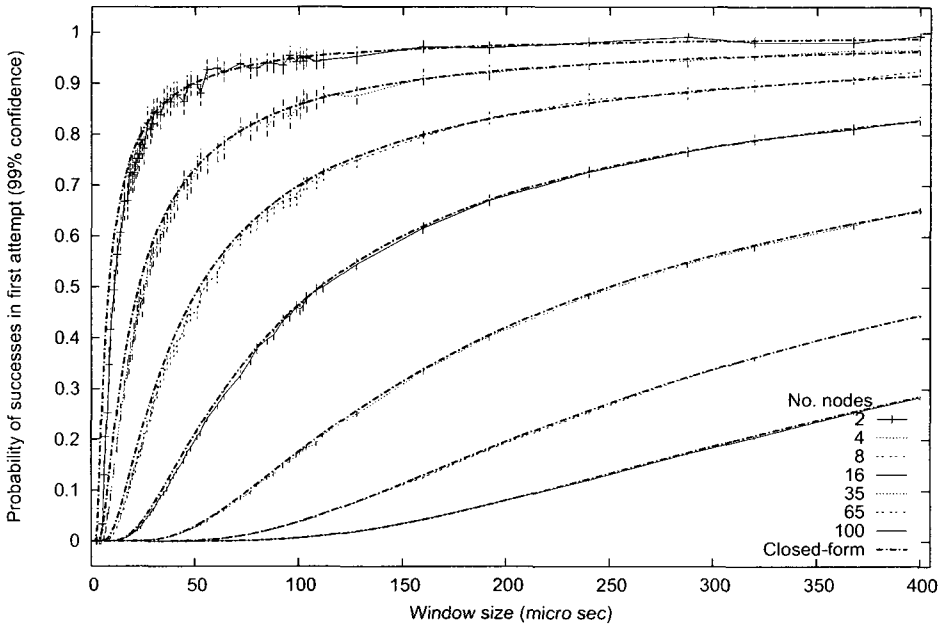


Figure 2-14: Comparison of the probability of success for n identically distanced nodes obtained from simulation and closed-form expression. Thick and thin lines show the value from the closed-form expression and simulation respectively in each curve. Vertical lines show 99% confidence intervals.

model matches the simulation precisely except for a small range of window sizes in the uniformly random case when the number of devices is very large. This error results from our assumption that the independence of two or more collision events. Our simulations show that the error introduced is negligible and is present only for a small range of window sizes. For window sizes larger than those appearing in Fig. 2-15, we have verified that the error diminishes rapidly.

2.5 Efficiency of the contention window

In the IEEE EPON registration scheme, the headend must reserve the communication link every time it needs to allow new nodes to register. Thus, a valuable portion of the available bandwidth is used at every such discovery cycle. The headend must reserve

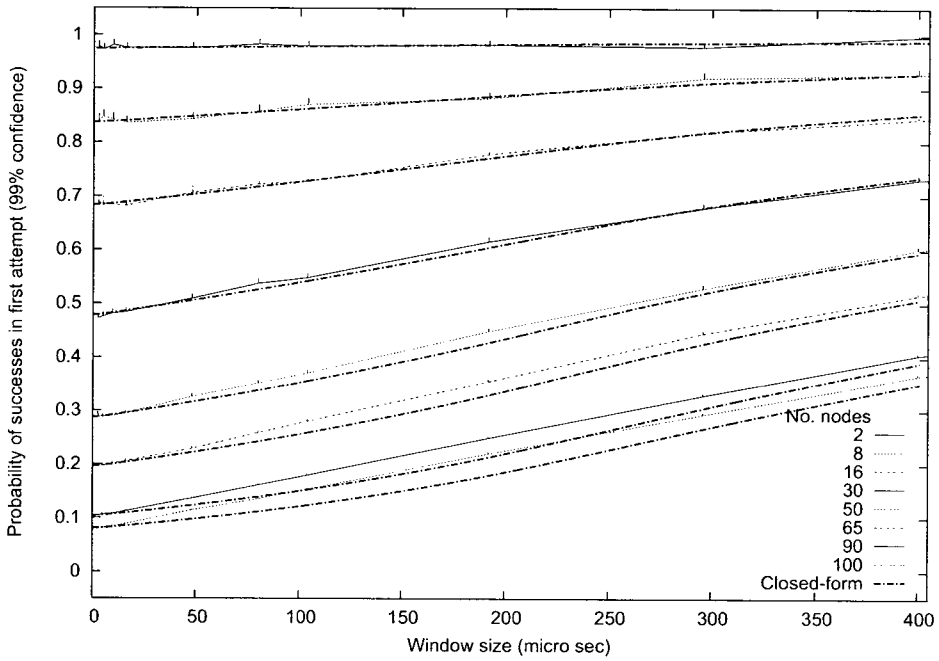


Figure 2-15: Comparison of Probability of Success for n uniformly randomly distanced nodes obtained from simulation and closed-form expression. Thick and thin lines show the value from the closed-form expression and simulation respectively in each curve. Vertical lines show 99% confidence intervals.

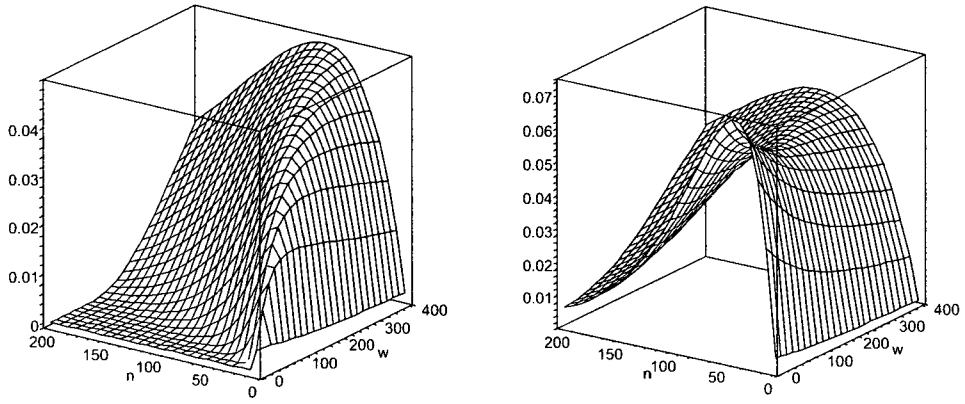


Figure 2-16: Contention window efficiency for n identically distanced *(left)* and randomly distanced *(right)* nodes.(w is in μs.)

the channel for a duration of $2p_{\max} + w$ where $p_{\max} = 100$ μs (20 km) as specified by the IEEE standard. It is desirable to minimize this duration when the channel is exclusively used for discovering new devices—regular traffic cannot be transmitted. To take this criterion into account, we can define a measure for the efficiency of a particular contention window size as the ratio of the average number of successful registrations to the size of the duration of the reservation [68]. Thus, efficiency

$$\rho = \frac{n \cdot P_s(n)}{2p_{\max} + w}. \tag{2.17}$$

We use our n-node model to relate efficiency to window size and node number. Fig. 2-16 shows the variation of efficiency with the window size and node number for the identically distanced (left) and the uniformly randomly distanced (right) cases. For the identically distanced case, Table 2.1 shows the most efficient window size for a given number of nodes, i.e., the smallest window size that maximizes the success probability. Due to the shape of the surface in Fig. 2-16 (right) equivalent maxima cannot be obtained for the uniformly random case. However, Fig. 2-17 shows the most efficient number of nodes that can be serviced by a contention window of a given

Table 2.1: The most efficient contention window size for identically distanced nodes.

No. devices ($p = 0$)	Most efficient window size (μs)
2	35.82
4	64.63
8	105.20
10	122.39
16	168.43
32	273.77
50	380.49
64	459.65
100	655.74
200	1179.31

size.

2.6 Summary and Future Work

We derived the probability of message collision in the 802.3ah EPON registration scheme. We derived an approximating closed-form expression for the probability of message collision in the 802.3ah EPON registration scheme. We compared the probability computed by the expression with simulation results and obtained a reasonably precise match. Further, we used our model to compute the most efficient contention window sizes for identically and randomly distributed nodes.

We are currently working on an exact solution of Eqn. (2.8). Approximate collision probabilities for smaller clusters of nodes and other distributions of nodes can be computed using the current model by setting the appropriate value for parameter p.

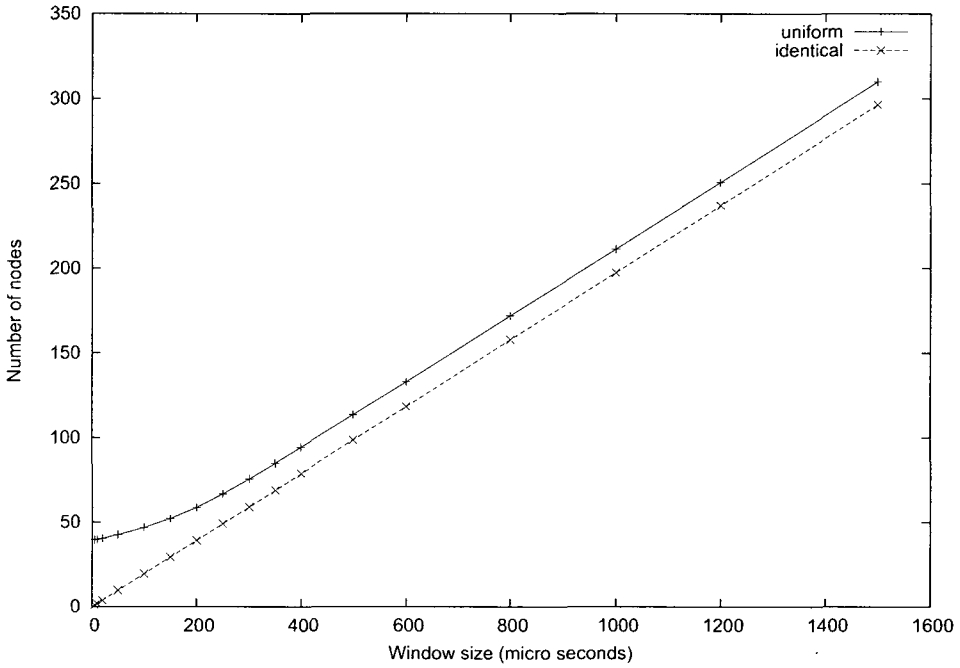


Figure 2-17: Number of nodes served by a window size with maximum efficiency.

However, multiple clusters cannot be modeled with the current setup. Now that the average number of successful registrations is known, the model can be extended to evaluate the multi-step performance of the registration scheme. Specifically, we can now model our scheme proposed in [22] and evaluate its efficacy.

Chapter 3

A Survey of Bandwidth Allocation Schemes for Ethernet Passive Optical Networks

3.1 Introduction

We present a comprehensive survey of the current state of research in the area of quality of service and bandwidth allocation for the recently standardized IEEE Ethernet Passive Optical Networks (EPONs). Our survey covers the majority of the work published in this area from the time the IEEE standard entered its final stages to the present time. This constitutes more than forty technical papers published in conferences and journals. Secondly, unlike other previous surveys, we fill in many of the details of the schemes surveyed which are missing in their respective original papers. We take a broad look at the direction of the work in this area as well as its relationship to past work in related areas. We also attempt to address, and in some cases provide solutions to, the shortcomings of proposed schemes that have been pointed out in subsequent literature. In parallel, we point out some new observations about the proposed schemes that have not been discussed in literature before. We hope that this survey will communicate the broad picture of the state-of-the-art in this area and its relationship to other theoretical and practical work.

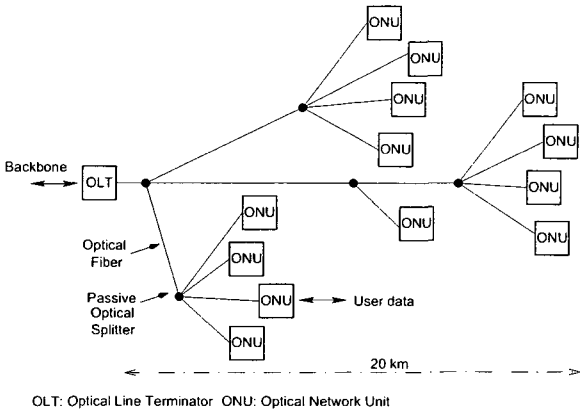


Figure 3-1: An Ethernet Passive Optical Network.

An *Ethernet Passive Optical Network (EPON)* is a point-to-multipoint, bidirectional, high rate optical network for providing subscribers with access to the backbone WAN. Fig. 3-1 shows a schematic diagram of an EPON. The EPON link is shared by multiple users. Each user connects to the EPON link through a device known as an *Optical Network Unit (ONU)*. Since the link is shared, scheduling of link use must be centrally arbitrated. This function is performed by a single special device called the *Optical Line Terminator (OLT)*. The direction of communication from the ONUs to the OLT is known as *upstream* direction whereas the direction from the OLT to the ONUs is known as the *downstream* direction. The data rate in each direction is set to 1 Gbps by the IEEE EPON standard [3]. Overall, the link exhibits a tree topology with the OLT at the root of the tree and the ONUs at the leaves. The EPON link is shared by all users in the upstream direction and is divided into discrete time slots. The OLT schedules which ONU is allowed to transmit data and for how many time slots. Appended to the data traffic, the ONU also transmits a control message containing a report of the number of bytes buffered in its queue, waiting for a transmission opportunity. Using these reports, the OLT is free to come up with a schedule that maximizes throughput, link utilization and minimizes the delay experienced by

user traffic. To enable the scheme described above, the IEEE standard [3] provides a minimal protocol known as the *Multi-Point Control Protocol* (MPCP). The MPCP consists of two types of messages:

1. The Report *Message* is sent by the ONU to the OLT. The ONU informs the OLT of the number of bytes waiting in its queue to be transmitted. The OLT uses the values reported in the Report message in order to come up with an efficient transmission schedule. Notice that the ONU requires a transmission opportunity to transmit the Report message as well. Typically, the ONU uses a small portion of any transmission opportunity granted to it by the OLT for transmitting the Report message. The IEEE standard [3] allows the OLT to require or prohibit the ONU from sending a Report message along with its transmission.

2. The Gate *Message* is sent by the OLT to the ONU to inform the ONU about its upstream transmission opportunity. The Gate message contains a transmission start time and transmission length. The ONU must begin transmission at the specified time and must only transmit the specified number of bytes (i.e., duration).

3.2 Survey Approach

There have been attempts in the recent past to survey the body of research in the area of EPONs [83, 120]. Many other papers provide an overview of work related to the focus of their paper. However, since their goal was to provide a simple, clear and concise introduction to the body of research, they do not have an opportunity to critique the schemes discussed in any detail or depth. Moreover, they only attempt to describe the work referenced but do not identify the key issues involved in this

area. These are the two points on which this survey differs from past attempts.

3.3 Assi et al.: LAER with Early Allocation [15]

Assi et al. propose a simple DBA scheme which we shall call the Limited Allocation with Excess Redistribution (LAER) [15]. They also propose an extension to the LAER algorithm which provides early allocation in conjunction with LAER for lightly loaded ONUs. We describe the LAER algorithm first and then the Early Allocation extension.

3.3.1 Description

The LAER algorithm operates on a cycle of fixed length (say, T_{cycle}). (Assume T_{cycle} to be the net number of data bits available after excluding all control overhead.) The grant size $g(i,j)$ alloted to any ONU j in a cycle comprises the sum of the grant size allocated by two procedures: Limited Allocation ($g_{la}(i,j)$) and Excess Redistribution ($g_{er}(i,j)$).

Limited Allocation

If the queue report received by the OLT in the previous cycle $i-1$ from ONU j is $q(i-1,j)$, then the Limited Allocation procedure allocates a grant of size

$$g_{la}(i,j) = \min(q(i-1,j), G_{min}(j)). \tag{3.1}$$

to ONU j in cycle i. The bound $G_{min}(j)$ of the limited allocation is calculated by dividing the total number of data bits that can be transmitted in a cycle among ONUs in proportion to the weight w_j assigned to each ONU j. The weights may be defined by an SLA such that $\sum_{j=1}^{N} w_j = 1$.

Excess Redistribution

Notice that,

$$\sum_{j=1}^{N} g_{la}(i,j) = T_{cycle} \tag{3.2}$$

if and only if $q(i-1,j) \geq G_{min}(j)$, i.e., under high load. For medium and low loads, the total cycle of duration T_{cycle} is not completely utilized. The objective of the Excess Redistribution procedure is to redistribute these excess transmission opportunities proportionally to needy ONUs. Let $L(i)$ and $H(i)$ denote set of lightly and heavily loaded ONUs respectively in cycle i and be defined as:

$$L(i) = \{j \mid q(i-1,j) \leq G_{min}(j)\} \tag{3.3}$$

$$H(i) = \{j \mid q(i-1,j) > G_{min}(j)\} \tag{3.4}$$

with $j \in \{1,\ldots,N\}$. Then, the Excess Redistribution procedure calculates the total number of bits available in a cycle from ONUs in $L(i)$ for redistribution to ONUs in $H(i)$ as

$$G_{er}(i) = \sum_{j \in L(i)} [G_{min}(j) - q(i-1,j)]. \tag{3.5}$$

These excess bits are redistributed to ONUs in $H(i)$ according to the following redistribution scheme:

$$g_{er}(i,j) = G_{er}(i) \cdot \frac{q(i-1,j)}{\sum_{h \in H(i)} q(i-1,h)}, \text{ if } j \in H(i), \tag{3.6}$$

$$= 0 \text{ otherwise.} \tag{3.7}$$

Finally, the total size of the grant allocated to an ONU j in cycle i by the LAER algorithm is

$$g(i,j) = g_{la}(i,j) + g_{er}(i,j). \tag{3.8}$$

The original paper also adapts the LAER algorithm to different classes of service (high, medium and low priority queues). It also proposes a simplistic prediction

scheme for estimating the amount of high priority traffic at the ONU at the time of its transmission opportunity. Briefly, the high priority traffic in the current cycle is predicted by the scheme to be same as that actually observed in the previous cycle [15].

Note that the LAER algorithm must collect Report messages from all the ONUs before it can begin computing the grant sizes for the next cycle. This may impose a delay comprising the LAER computation time and the RTT to the first ONU to be polled in the next cycle during which the EPON link will remain idle. To remedy this underutilization, the paper proposes an Early Allocation extension to the LAER algorithm which is as follows. On receiving a Report message $q(i-1, j)$ from an ONU j, the OLT checks whether $j \in L(i)$, i.e., whether ONU j is a lightly loaded ONU. If so, then the OLT schedules a grant for ONU j at the next earliest possible time. For all other ONUs $j \in H(i)$, the OLT transmits a Gate message to them only after it has collected Report messages from all ONUs. In effect, the EPON link could possibly be utilized by the lightly loaded ONU transmissions during the LAER computation time and the RTT to the first heavily loaded ONU. A concise pseudocode of this LAER with Early Allocation algorithm is provided in Fig. 3-2.

3.3.2 Discussion

The simulation results reported by the authors [15] show the superior performance of the LAER algorithm over a static bandwidth allocation (SBA) scheme and the superiority of LAER+EA over LAER. The particular values for the weights w_j used in the simulation are not reported. Notice that when comparing with SBA, the fixed grant size assigned to any ONU under the SBA scheme should be chosen carefully. For example, if the LAER algorithm is to be compared against the SBA scheme, then the size of the grant assigned to any ONU j by the SBA scheme should be chosen

ALGORITHM LAER+EA($\{w_1, \ldots, w_N\}, T_{cycle}$)

1 Calculate $G_{min}(j)$
2 **for** each cycle i **do**
3 $rc \leftarrow 0$
4 **repeat**
5 Receive report $q(i-1, j)$ from ONU j
6 $rc \leftarrow rc + 1$
7 Set $g_{la}(i, j) \leftarrow \min(q(i-1, j), G_{min}(j))$
8 **if** $q(i-1, j) \leq G_{min}(j)$
9 **then** $L(i) \leftarrow L(i) \cup \{j\}$
10 Send grant of size $g_{la}(i, j)$
11 **else** $H(i) \leftarrow H(i) \cup \{j\}$
12 **until** $rc = N$
13 Calculate $G_{er}(i)$
14 **for** each ONU $j \in H(i)$ **do**
15 Calculate $g_{er}(i, j)$
16 Send grant of size $g_{la}(i, j) + g_{er}(i, j)$

Figure 3-2: The LAER+EA Algorithm [15].

to be $G_{\min}(j)$ to ensure a fair comparison. The fixed grant sizes used for the SBA scheme are not reported in the paper.

The difference between the LAER and the LAER+EA algorithm is that the OLT allocates a grant to a lightly loaded ONU as soon as it receives a Report from that ONU whereas in the LAER scheme, all ONUs must wait until all Report messages have been collected. Clearly, the delay under the LAER+EA scheme will always be less than or equal to that under the LAER scheme. Thus, it is safe to say that given the LAER+EA scheme, the LAER scheme is undesirable.

Since the the performance of the LAER+EA scheme is not compared against any competing schemes except for the naive SBA scheme, it is difficult to judge the advantages of the LAER+EA scheme. One desirable feature of the LAER+EA scheme is its ability to guarantee a minimum grant size ($G_{min}(j)$) and therefore a minimum bandwidth to any ONU j. Another advantage of the scheme is that regardless of the weight, any overloaded ONU j can utilize the bandwidth unused by other ONUs. Moreover, the amount of this extra bandwidth available to such an overloaded ONU is proportional to its fraction of the total demand (see Eqn. (3.6)). However, observe that as per Eqn. (3.6) an ONU $j \in H(i)$ may receive a grant larger than its request, since the excess bandwidth is divided only in proportion to the request without considering the actual size $q(i-1,j)$ of the request itself. This may lead to some underutilization. Recently, Shami et al. [103] proposed a simple alternative redistribution policy to eliminate the possibility of allocating a grant larger than requested. According to their policy, if the total demand

$$\sum_{j=1}^{N} q(i-1,j) \leq T_{cycle} \tag{3.9}$$

then, since the demand from all ONUs can be satisfied, they suggest that

$$g(i,j) = q(i-1,j). \tag{3.10}$$

Only if the total demand is larger than the available grant time is the Excess Redistribution procedure of Eqn. (3.6) used. In this latter case, since

$$\begin{aligned}
\sum_{j=1}^{n} q(i-1,j) &> \sum_{j=1}^{n} G_{min}(j), \text{ therefore,} \\
\sum_{j=1}^{n} q(i-1,j) - G_{min}(j) &> 0 \text{ and therefore,} \\
\sum_{h \in H(i)} q(i-1,h) - G_{min}(h) &> \sum_{l \in L(i)} G_{min}(l) - q(i-1,l) \\
\sum_{h \in H(i)} q(i-1,h) - g_{la}(i,h) &> G_{er}(i)
\end{aligned}$$

is guaranteed, and hence,

$$\frac{q(i-1,j) - g_{la}(i,j)}{\sum_{h \in H(i)} q(i-1,h) - g_{la}(i,h)} < \frac{q(i-1,j) - g_{la}(i,j)}{G_{er}(i)} \tag{3.11}$$

holds for all ONUs $j \in H(i)$. Hence, the excess redistributed to any ONU is smaller than the excess requested by the ONU.

However, note the difference between the redistribution policy of Eqn. (3.11) and that of Eqn. (3.6). In Eqn. (3.11), the excess is redistributed according to the excess requested, i.e., $q(i-1,j) - g_{la}(i,j)$ and not the total request $q(i-1,j)$ of any ONU $j \in H(i)$. This is crucial to the "no overgranting" claim by Shami et al. [103]. If the excess is redistributed as in Eqn. (3.6) according to the total request $q(i-1,j)$, then it is easy to show that the modification suggested by Shami et al. does *not* guarantee that each ONU is granted no more than it requests and the claim by Shami et al. does not hold. To see this, suppose $G_{er}(i) = 4G$ and consider two ONUs k, m such that

$$\begin{aligned}
q(i-1,k) &= G_{min}(k) + 4G \\
q(i-1,m) &= G_{min}(m) + G, \text{ and} \\
G_{min}(m) &= 4G_{min}(k) = 4G
\end{aligned}$$

for some fixed $G > 0$. Observe that the demand condition required by Shami et al. [103] is satisfied since $G_{er}(i) < 5G$. Yet, if the excess is now redistributed according to Eqn. (3.6) as suggested by Shami et al. [103], then

$$g_{er}(i,m) = G_{er}(i) \cdot \frac{5G}{10G} = 2G \tag{3.12}$$

Hence,

$$g(i,m) = 2G + G_{min}(m) = 6G > q(i-1,m). \tag{3.13}$$

Thus, it is important that the excess be redistributed in proportion to only the excess requested by any ONU j according to Eqn. (3.11) and not in proportion to the total request of ONU $j \in H(i)$ if no overgranting is to be guaranteed for the LAER algorithm.

In general, the LAER scheme inherits the desirable properties of a Generalized Processor Sharing (GPS) server. A GPS server capable of service at a constant rate r is defined [90] as one in which the amount of service provided to any session (task, or ONU) j with a positive queue length in a time interval of length T is:

$$S_j(T) \geq \frac{\phi_j}{\sum_k \phi_k} rT. \tag{3.14}$$

In each cycle of length T_{cycle}, the LAER algorithm serves an ONU j exactly as per the GPS rule above under heavy load (i.e. when $L(i) = \emptyset$). It is well known that a Generalized Processor Sharing server allocates the processor most fairly among all tasks (i.e., ONUs). The smaller the cycle length, the more faithful will be the simulation of the GPS scheme by the LAER algorithm.

One way to view the motivation behind the design of the LAER algorithm is to see it as an adaptive extension of the SBA scheme. (The authors compare LAER to SBA in the paper and we speculate that the reason for doing so may be related to the motivation behind the LAER design.) The main drawback of the SBA scheme is its insensitivity to changing load at an ONU leading to underutilization of the link.

LAER alleviates this shortcoming by allowing granted slots in each cycle to adapt to the changing load proportionally. However, the cycle length remains fixed. Therefore, if the total load on the EPON were to change, then the fixed cycle length would create the same problem in LAER as does the fixed grant size in the SBA scheme. The LAER algorithm takes a Processor Sharing approach to dividing the cycle of fixed length. In each cycle, it provides some quantum of service to each ONU. Within the cycle, the granted quanta are adaptively sized to service lightly and heavily loaded ONUs. However, if the load changes, then the cycle may become too small to accommodate the new load. Consider the limiting case where only one bit from each ONU can be served in a cycle. Thus, the cycle length will be N bits for N ONUs. For identically loaded ONUs each with a queue length of q bits, the average completion time would be approximately Nq. Now consider if the cycle time were increased to be Nq. The completion time for the first ONU would be q, the next $2q$ and so on. The average completion time would be $q(N+1)/2$. All intermediate increasing cycle lengths from N to Nq will provide increasingly better delay performance. Thus, the cycle length must be chosen carefully in relation to the load. While a Processor Sharing approach provides fairness by delaying each "job" due to all other jobs equally, it by no means provides an optimal delay schedule.

Next, note that the LAER+EA scheme waits until it has received complete information about the queue lengths of all the heavily loaded ONUs. Under such conditions, allocating grants in increasing order of queue length will always yield a *more optimal schedule.* Thus, the LAER+EA scheme ought to allocate grants to heavily loaded ONUs in increasing order of their reported queue lengths. This simple rule can improve the performance of any scheme that waits to receive queue length information from all ONUs before constructing a schedule.

3.4 Zheng's Idling-Avoidance Scheme

Zheng et al. [121] propose a simple modification to the basic LAER+EA algorithm. Their proposal attempts to eliminate the channel idling in the LAER+EA scheme when the OLT examines computes a fair share for each heavily loaded ONU and transmits a Gate message to each ONU $j \in H(i)$ (see Eqn. 3.4).

3.4.1 Description

Recall that in LAER+EA, an ONU $j \in L(i)$ is granted a transmission opportunity immediately (i.e., a Gate message containing that opportunity is transmitted to ONU j immediately) whereas all other ONUs $k \in H(i)$ are only serviced at the end of the current cycle i. This allows the OLT to collect Report messages from all ONUs $j \in H(i)$ and calculate the fair share of the excess bandwidth that should be allocated to each ONU $j \in H(i)$. Suppose cycle i ends at time t. If the OLT takes time $t_{LAER+EA}$ to finish its DBA computation (i.e., execution of the LAER+EA fair apportionment computation from Eqn. 3.6), and if the RTT to the first ONU $j \in H(i)$ to be scheduled is t_j, then the channel cannot be used for data transmission in the interval $[t, t + t_{LAER+EA} + t_j]$. The goal of the proposal by Zheng et al. is to fill this "hole" in the schedule. The proposed scheme works as follows. Let t_{RTT} be the maximum RTT. Upto time $t - t_{RTT}$, the basic LAER+EA scheme is followed. Note that time $t - t_{RTT}$ is the latest time at which a grant can be scheduled with no channel idling. Therefore, at this time, the OLT checks to see if there is any potential for channel idling. For example, due to early allocation to some ONUs $j \in L(i)$, the "hole" may have been completely filled and thus idling may have been eliminated already. If so, the original LAER+EA algorithm is followed. If not, then an ONU $j \in H(i)$ is scheduled early to help fill the "hole" in the schedule. This is done until the idle time is completely eliminated.

3.4.2 Discussion

The above scheme is effective only in the high load region since in this region it is likely that $L(i) = \emptyset$. In Zheng's proposal, since an ONU $j \in H(i)$ is scheduled early, its share of the excess bandwidth cannot be determined (accurately) at the time of its early allocation. Hence, it appears that this ONU $j \in H(i)$ may not be allocated its excess share resulting in a somewhat unfair allocation of bandwidth as compared to the original LAER+EA scheme. The unfairness will be the greatest in the case where the ONU selected for early allocation is one that would have received the largest share of the excess bandwidth.

This unfairness could be at least minimized if the ONU $j \in H(i)$ chosen for early allocation is the one with the smallest demand among other ONUs in $H(i)$. Alternatively, another grant could be sent to this ONU to provide it with its share of the redistributed excess bandwidth. However, the gain of this solution must be weighed against the bandwidth that will be wasted in the extra guard time required for the new grant.

3.5 Bai, Shami et al.: Hybrid Granting [18][102]

Bai, Shami et al. propose the Hybrid Granting (HG) scheme to provide low delay as well as low jitter (delay variance) [18][102]. They follow the general approach of LAER algorithm proposed by Assi et al. [15] in that HG is also based on the idea of apportioning the total bandwidth according to fixed weights first and then redistributing the excess bandwidth according to excess demands. However, HG decouples grants for high priority traffic to all ONUs from those for traffic of a lower priority. The advantage of this separation is that the variability in low priority traffic demand cannot affect the delay for a high priority traffic grant, thus reducing the jitter for high priority traffic.

3.5.1 Description

The HG scheme divides a maximum time-frame T_{cl}^{max} into two portions: one in which grants are allocated to expeditiously forwarded (EF) (high priority) traffic from every ONU and the other in which grants are allocated to assuredly forwarded (AF/BE) (lower priority including best effort traffic) traffic from all ONUs. The design is based on the following two (arguably valid) assumptions. 1) EF traffic is quite predictable and the bandwidth required (in terms of bits/second) to service it is exactly known in advance. 2) AF/BE traffic on the other hand is bursty, unpredictable and requires "online" allocation decision using the information obtained from Report messages. Under the HG scheme, the AF grant portion of the total time-frame follows the EF grant portion. That is, the EF grants to all ONUs are scheduled first followed by the AF grants to all ONUs. An ONU transmits a Report message indicating the amount of accumulated AF/BE traffic during its allocated EF grant. After the OLT has collected this information from all the ONUs, it calculates the AF share of each ONU and sends out AF/BE grants to ONUs. The AF portion in each time-frame thus begins after a delay equal to the maximum RTT from the end of the EF portion of the window. Also note that the size of each time-frame, although bounded, may be variable. Hence, the size of the EF grant to any ONU in each time-frame may vary. The HG scheme calculates the EF grant size for ONU j as follows. If the the ith EF grant $g_{EF}(i,j)$ to ONU j was scheduled at $t_{EF}(i,j)$, then

$$g_{EF}(i+1,j) = (t_{EF}(i+1,j) - t_{EF}(i,j)) \cdot R_{EF}(j) + r, \tag{3.15}$$

where $R_{EF}(j)$ is the bit-rate of the EF traffic that is to be reserved for ONU j and is known in advance and r is the length of a report message. The HG scheme calculates the AF/BE grant size using a method similar to the LAER algorithm (see [102] or [18] for details).

3.5.2 Discussion

The HG scheme is largely based on the same idea as LAER [15]. However, HG decouples the bursty and unpredictable AF/BE traffic grants from the predictable EF grants. The effect of this separation on the delay and delay variance of the EF traffic seems to be substantial. Since the starting time of the EF grants are now distributed in a smaller range, their variance (and hence the delay variance) is smaller.

There have been two proposals to improve the HG scheme. The first proposal by Kim et al. [66, 65] is concerned with the problem of channel idling in the HG scheme. Under the HG scheme, the EF portion of the cycle precedes the AF portion. Moreover, the Report messages required to calculate the AF grant apportionment are only received in the EF cycle. Thus, between the EF and the AF cycle, there exists a period of time equal to the RTT to the ONUs during which the channel is idle. The HG scheme needs this time to calculate AF grant sizes and transmit the AF grants to the ONUs, and wait for the first AF/BE data bit to reach the OLT. Kim et al. [66, 65] argue that Report messages could be collected in the previous AF cycle rather than the current EF cycle. In this way, the necessary Report messages would be available way ahead of time and the channel idle time could be avoided. Shami et al. [102] do address this problem. In Sec. III (c) and Fig. 2 in [102], Shami et al. argue that Report messages collected in the previous cycle are less recent than those collected in the current EF cycle. Hence, grants for the bursty AF/BE traffic based on stale information from Report messages received in the previous cycle may consistently underestimate the traffic at the ONU. This may have the undesirable consequence of some AF/BE traffic either being dropped or considerably delayed. Preliminary results in [66, 65] indicate that the effect of stale Report messages on AF/BE traffic may be less significant compared to the gain in channel utilization due to a reduction in channel idling. Another idea for improving the HG scheme is to eliminate the

variability in the size of the EF grant allocated so as to reduce intra-grant jitter.

3.6 Bai et al.: Fair LAER

Fairness of bandwidth allocation is an important issue that has received relatively little attention. Recently Bai et al. [16][17] have discussed the fairness of the LAER algorithm and have proposed an alternative which we shall refer to as Fair-LAER (FLAER).

3.6.1 Description

As described in Sec. 3.3.1, the LAER algorithm redistributes excess transmission opportunities in each cycle in proportion to the demands of heavily loaded ONUs $j \in H(i)$. Notice that if a single ONU j were to report a large queue $q(i-1,j) > G_{min}(j)$, then as per the LAER algorithm, a large fraction of the excess bits $G_{er}(i)$ would be allocated solely to ONU j. Bai et al. [16][17] argue that such behavior may potentially starve many less heavily loaded ONUs in favor of a few very heavily loaded ONUs and is therefore unfair to the less heavily loaded ONUs. They propose that the fair share of the excess bandwidth for an ONU j ought to be calculated using the fixed weight w_j of the ONU and not its current demand (see Eqn. 3.6). In this way, fairness in sharing excess bandwidth can be guaranteed in the absolute sense (i.e., w.r.t. weights w_j) rather than in the relative, dynamic sense (i.e., w.r.t. current demands). We refer the reader to [16][17] for details.

3.7 Ma et al.: Bandwidth Guaranteed Polling [81]

One of the earliest proposed schemes to provide bandwidth guarantees for ONUs is the Bandwidth Guaranteed Polling (BGP) algorithm [81]. We provide a brief description

of the algorithm below; we also refer the reader to the previous survey for a concise and highly accessible description of the BGP scheme [83].

3.7.1 Description

One of the main motivations behind the design of the BGP scheme is the provision of bandwidth guarantees to premium subscribers. As proposed in the original paper, the BGP scheme groups ONUs into two classes: bandwidth-guaranteed ONUs (class 1) and non-bandwidth guaranteed ONUs (class 2). Bandwidth is explicitly reserved for the premium subscriber-ONUs in the first group whereas ONUs in the second group only receive any available bandwidth unused by class 1 ONUs. The satisfaction of the bandwidth demand of a class 1 ONU is based on two key elements of the BGP scheme. First, the BGP scheme maintains a polling table which is a schedule of class 1 ONUs (i.e., a sequence of ONU identifiers) to be polled in a single cycle of the BGP algorithm. A polling sequence is also constructed for class 2 ONUs, but here each class 2 ONU appears only once in the list. In each cycle, the BGP algorithm iterates over the entries in the class 1 polling table. If the current entry is for ONU i, then the algorithm generates a grant message for ONU i. Note that the polling table may have multiple entries for ONU i allowing ONU i to be granted multiple grants in the same cycle. This is the first mechanism by which the BGP algorithm controls the bandwidth allocated to any class 1 ONU. The second mechanism is the size of the grant allocated to the ONU. This is a parameter W_{max} set by the operator. Every grant sent to a class 1 ONU is of size W_{max}. By choosing suitable values for W_{max} and the length of the polling table for a single cycle, any feasible set of bandwidth demands for class 1 ONUs can be satisfied to within a close approximation. This defines the behavior of the BGP algorithm with respect to class 1 ONUs.

Recall that class 2 ONUs by definition are allowed to receive only excess, available

bandwidth, if any. The BGP scheme implements this definition in the following way. Whenever a class 1 ONU receives a grant it informs the OLT of the actual size of its impending transmission using the Report message. (Note that the BGP scheme is different in this regard from other schemes since according to BGP, the ONU sends a Report message before its data transmission. We discuss this issue later.) The ONU then continues with the transmission of the queued data, if any. On receiving such a Report, the OLT first checks if the portion of the grant used by the current ONU is zero (i.e., the current class 1 ONU has no data to transmit at this time). If so, the OLT polls the next class 1 (not class 2) ONU immediately. However, if the size of the grant used by the current ONU is non-zero, then the OLT decides if the size of the unused portion of the grant (if any) is large enough to be reassigned to a class 2 ONU. Clearly, this decision must take into account the time required to send a grant to and receive data from the next class 2 ONU. In other words, the superfluous portion of the grant must be larger than the RTT to the class 2 ONU intended to use it. However, the BGP scheme relaxes this requirement. Instead of the stricter RTT rule mentioned above, it uses the following weaker but slightly simpler threshold rule. If the size of the unused grant is larger than a fixed threshold T, then the BGP algorithm decides to reassign the superfluous grant to the next class 2 ONU in the class 2 polling list. If the superfluous grant is not larger than the threshold, then the BGP scheme chooses not to reassign the unused grant, in which case the grant remains underutilized.

The above rules concerning reassignment of superfluous grants are applied only after receiving a Report from a class 1 ONU. Thus, the BGP scheme does not apply the superfluous reuse rules to the Report received from a class 2 ONU, i.e., if the class 2 ONU does not use all of the superfluous grant reassigned to it, no more reassignments occur. Instead, the next class 1 ONU is polled and the BGP algorithm

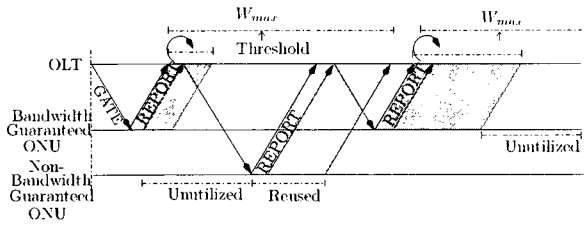


Figure 3-3: An illustration of the BGP grant reassignment scheme with one bandwidth guaranteed and one non-bandwidth guaranteed ONU.

continues as discussed above. Similarly, if the superfluous portion recovered from a class 1 ONU is smaller than the threshold (i.e., too small to be reassigned) or is equal to the grant size (i.e., the grant will not be utilized by the current class 1 ONU at all), then the next class 1 ONU is polled and the BGP algorithm continues as above until it reaches the end of the polling table after which it starts a new polling cycle. Thus, in each case, the next ONU polled is picked from the class 1 polling table. Finally, the BGP algorithm also mentions that unused entries in the class 1 polling table may be used to poll class 2 ONUs without any further details about this step of the algorithm.

3.7.2 Discussion

Clearly, the BGP algorithm uses the Report message in a way different than other schemes. (Particularly, we focus on the IPACT scheme for comparison, due to its relative simplicity and since the original paper compares its performance to IPACT as well.) Firstly, as per the definition of the BGP scheduling algorithm, the contents of the Report message depend on the allocated grant size in the following way: If the allocated grant is G, then the reported length is $\min\{Q, G\}$. Clearly, if the size of the ONU buffer is $Q > G$, then this valuable information is lost as it is not conveyed to the OLT as per the definition of the BGP algorithm. While this may not be a good design guideline in general for EPON DBA schemes, for the BGP algorithm, this may

not be as serious a problem. An argument can be made that the BGP algorithm does not and perhaps cannot benefit from this information without major modifications to the BGP scheduling algorithm. Thus, losing that information makes no difference. On the other hand, allowing the ONU to report its actual buffer length Q instead of $\min\{Q, G\}$ allows all of the available information to be conveyed to the OLT without affecting any other step in the BGP algorithm. The OLT can easily figure out the fraction of the grant that the ONU intends to use by tracking the grant size offered to the ONU.

Second, unlike IPACT, in BGP, the Report message is transmitted by the ONU prior to its data transmission. Third, The OLT always allocates grants of size W_{max} and it is the ONU which informs the OLT of the fraction of the allocated grant that it will use for its current transmission. This allows the OLT to recover any superfluous grant time and reassign it to a lower class ONU. Finally, the polling table allows the BGP scheme to design a flexible scheduling cycle where an ONU may be serviced more than once in the same cycle. Thus, within a single cycle, depending on the distribution of an ONU's polling positions in the table, a variety of different delay distributions can be provided to different ONUs. If the polling entries are distributed evenly, as is accomplished by the Even Distribution Algorithm (EDA) in the BGP scheme, then some fixed delay can be guaranteed. The polling table provides a flexible way to provide any demanded delay guarantee to within a close approximation.

Now consider the following (novel) modifications to the original IPACT scheme under gated, limited allocation service discipline with maximum grant limited to W_{max} [71]. Suppose that the given ONUs belong to two different sets, class 1 and class 2. Further suppose that instead of the round-robin schedule, the IPACT scheme follows the polling sequence obtained by concatenating the polling sequences from the BGP polling table for class 1 followed by class 2. Finally, assume that the class

2 ONUs are served exactly as per the IPACT algorithm except that the maximum limit for their grant size in cycle i, call it $W'_{max}(i)$ is calculated in such a way that the total cycle time is bounded. For example, if there are N_1 class 1 ONUs and N_2 class 2 ONUs, then for cycle i,

$$W'_{max}(i) = \frac{1}{N_2}\left[N_1 W_{max} - \sum_{j=1}^{N_1} g(i,j) - \max_{1\leq k\leq N_2} \frac{d_k}{\delta}\right], \tag{3.16}$$

where $g(i,j)$ is the grant allocated by IPACT to ONU j in cycle i, $2d_k$ is the RTT to ONU k and δ is the time required to transmit a single bit on the EPON link (i.e., 10^{-9} seconds). Then, the total cycle time

$$T_{cycle} \leq \delta N_1 W_{max}. \tag{3.17}$$

This modified IPACT scheme can, on an average:

1. guarantee the same demand set to class 1 ONUs as the BGP scheme,
2. provide the same fraction of traffic to class 2 ONUs in each cycle as BGP (with some added delay),
3. but, unlike the BGP scheme, with better throughput due to less bandwidth wasted during walk times.

One feature that makes BGP algorithm superior to the modified IPACT scheme is its ability to elicit the latest possible queue length information from the ONU. By its definition, IPACT cannot compete with BGP in this area (unless some traffic estimation module is added to IPACT). However, as discussed above, BGP can only make limited use of this latest information. Moreover, depending on the instantaneous load at the ONU and the RTT to the next ONU in the polling table, BGP may end up wasting valuable bandwidth in walk time. To be precise, if a cycle polls M ONUs and if the RTT to the next ONU to be polled is $2d_j$, then to achieve high utilization (i.e.,

back-to-back transmissions from consecutively scheduled ONUs) the original IPACT scheme requires that

$$2d_j \leq \delta(M-1)W_{max}, \tag{3.18}$$

where δ is the time required to transmit a single bit on the EPON link. This is a weaker requirement compared to the condition required by the BGP scheduling algorithm which is

$$2d_j \leq \delta W_{max}. \tag{3.19}$$

The only real performance benefit for the BGP scheme comes in the form of lower delay and only from its novel use of the Report message. If BGP were to use the Report message in the standard way, it would reduce to the modified IPACT algorithm characterized above. Thus, the polling table does *not* empower the BGP algorithm with any new capability over the IPACT algorithm (since the table can be split at its period and used by IPACT as its polling sequence). In fact, comparing the delay performance of the 1-entry ONU in the BGP paper [81] with IPACT illustrates this point: the polling table only makes the cycle longer whereas the same ONU is served once every (shorter) cycle under IPACT.

Notwithstanding the above discussion, BGP's novel use of the Report message does improve delay performance in comparison to IPACT (unless IPACT is empowered with an estimation algorithm). Except under low loads, the difference in bandwidth wasted due to walk time by the BGP scheme and the IPACT algorithm will be minimal with BGP delivering better delay performance than IPACT. However, BGP's best feature also happens to be its biggest drawback. The IEEE EPON standard [3] stipulates a model in which the OLT, through Gate messages, dictates the times when the ONUs must turn their lasers on and off. Following this model is mandatory for any DBA scheme to be compliant with the IEEE standard. In contrast, BGP allows the ONU to inform the OLT when it will turn off its laser through the Report message

conveying to the OLT the fraction of the grant it will use. Moreover, the reassignment of superfluous grants implies that the OLT may send out seemingly conflicting grants. This approach contradicts the stipulation of the IEEE standard and, as pointed out by others [15] may be considered incompatible with the standard—a fundamental shortcoming for any DBA scheme.

The performance results reported about the BGP algorithm [81] are based on simulation and analysis of Poisson traffic with fixed packet lengths and exponentially distributed interarrival intervals. Overall, BGP seems to provide delay performance better than IPACT for bandwidth guaranteed ONUs with at least four entries in the polling table. For the ONU with a single polling entry, the delay offered by IPACT is much better than BGP since the BGP cycle is longer than the IPACT cycle, as discussed earlier. The throughput obtained by the BGP scheme is lower than that obtained from the IPACT scheme. This is due to the increased bandwidth wasted during walk time during grant reuse. Thus, the BGP scheme trades a lower value of throughput for better delay performance.

3.8 Ma et al.: Adaptive Scheduling for Differentiated Services [80]

Recently, Ma et al. [80] proposed enhancements to the original IPACT scheme aimed at providing better delay guarantees for high priority traffic as well as improving the delay and throughput for traffic with lower priority under unbalanced traffic. Below we provide a description of the proposed enhancements and then point out some of our observations.

3.8.1 Description

The recent proposal by Ma et al. [80] proposes a scheme for improving IPACT performance under unbalanced traffic. It proposes a separation of the strategy for scheduling among ONUs at the OLT (i.e., inter-ONU scheduling) from the scheduling strategy at the ONU among its various traffic classes or queues (intra-ONU scheduling). The scheme replaces IPACT's round-robin inter-ONU scheduling [71] with a "Dynamic Polling Order arrangement" (DPOA) and IPACT's (strict) priority queue scheme [70] with a (weaker) "priority insertion scheduling" scheme.

The scheme also proposes polling only those ONUs in any cycle i that fall into the Selective Polling Set (SPS) $S(i)$ defined as follows:

$$S(i) = \{j \mid q(i,j) \geq (1-\alpha) \cdot \max_{1 \leq j \leq N} q(i,j)\}, \tag{3.20}$$

where $0 \leq \alpha \leq 1$ and $q(i,j)$ is the length of the queue of ONU j in scheduling cycle i. In other words, only the highly loaded ONUs are polled in cycle i with the load at any ONU being measured relative to the that of the ONU with the highest load in that cycle. Although not addressed in the original paper, for completeness, we note that according to this strategy, ONUs not in $S(i)$ are still polled in each cycle, but with grants only large enough to transmit a Report message. In the experiments reported in the original paper, $\alpha = 0.2$ [80].

The inter-ONU DPOA scheme proposes that in each cycle, the ONUs in the SPS be polled in decreasing order of their queue lengths. The polling order for the next cycle i is determined by sorting the queue lengths of the ONUs in the SPS $S(i)$ in descending order.

The proposed intra-ONU scheduling algorithm called Priority with Insertion scheduling (PIS) assumes the presence of a set of N_1 real-time (high priority) and N_2 non real-time (low priority) queues. Under the same conditions, IPACT uses the strict priority scheme where a packet from a low priority queue can be transmitted only

after the high priority queue has been served. As a result, low priority traffic incurs large delays, even under light loads [70, 15]. The PIS scheme attempts to improve the waiting time of low priority traffic without any significant effect on the high priority traffic. It assumes that a delay bound d_k for each packet k in the real-time queues is available. Whenever a new packet k with delay bound d_k enters a real-time queue, it is inserted at a position into the queue according to its urgency value such that the queue remains sorted in increasing order of urgency. At time t, the urgency of a packet k of length L_k with delay bound d_k that has waited at the ONU in queue j for time $w_k(t)$ is defined as

$$U_k^j(t) = d_k - w_k(t) - L_k. \tag{3.21}$$

To insert a new packet into the queue at time t, the ONU must be able to calculate the urgency of any packet in the queue at time t. A simple way to enable this calculation is for the ONU to save the time of arrival of the previous packet t_p (or, in other words, the time of previous calculation of the urgency). At current time t, the new value of urgency for each packet in the queue can be obtained as follows:

$$\begin{aligned} U_k^j(t) &= d_k - L_k - w_k(t) \\ &= d_k - L_k - [w_k(t_p) + (t - t_p)] \\ &= U_k^j(t_p) - (t - t_p). \end{aligned} \tag{3.22}$$

In general, the new value of urgency after time Δ has elapsed is:

$$\begin{aligned} U_k^j(t + \Delta) &= d_k - L_k - [w_k(t) + \Delta] \\ &= U_k^j(t) - \Delta. \end{aligned} \tag{3.23}$$

Thus, if a packet of length L_k is dequeued from a queue j at time t and transmitted, then the new urgency value after the transmission will be

$$U_k^j(t + L_k) = U_k^j(t) - L_k. \tag{3.24}$$

Given such a queue sorted in increasing order of urgency, the PIS scheme uses the following rules for choosing the next packet to transmit within a transmission grant:

1. If, and as long as the length of the high priority queues exceeds a fixed threshold Q_H, follow the strict priority scheduling among high priority queues.

2. Let $u^j = \min_{1 \leq j \leq N_1} U_0^j(t)$. Let L_0^{nr} be the length of the packet at the head of the (first) non real-time queue. Then:

   (a) If $u^j < 0$, then drop the packet at the head of queue j since its delay bound has been violated.

   (b) If $0 \leq u^j \leq L_0^{nr}$, then transmit the packet at the head of queue j since its delay bound would be violated if we were to transmit the packet at the head of the non real time traffic queue. Otherwise, transmit the packet at the head of the non real-time traffic queue.

Using these rules to choose the next packet, the PIS scheme offers better delay performance to non real-time traffic without drastically affecting the delay performance of real-time traffic.

3.8.2 Discussion

Recall that the Selective Polling Set (SPS) scheme chooses to grant data bandwidth only to the most heavily loaded ONUs as characterized by α. It can provide higher bandwidth for the heavily loaded ONUs in the SPS thus adapting to the current, asymmetric load conditions for small values of α. However, this can also have unwanted side-effects. For small values of α, especially under unbalanced traffic, a lightly loaded ONU j could be denied membership of the polling set $S(i)$ for many scheduling cycles because of other heavily loaded ONUs and could thus face starvation and high delay. Moreover, if the lightly loaded ONU carries high priority, real-time traffic (i.e.,

voice or video) with tight delay bounds, then the SPS strategy will provide poor, unfair performance to the lightly loaded ONU. Although the paper reports results separately for each of its two scheduling strategies, no results are provided to justify the effectiveness of this SPS scheme.

Under unbalanced traffic, the DPOA strategy can minimize the packet loss at heavily loaded ONUs and thus increase the overall throughput. Results reported in the paper show that the DPOA scheme combined with IPACT shows better throughput than IPACT alone. However, the details of how the scheme is combined with IPACT are not provided. IPACT is free to schedule a transmission from any ONU j as soon as it receives a Report message from the ONU. However, the DPOA scheme must wait until it receives the Report messages from all ONUs. Only then can it sort them, construct the SPS and start sending Gate messages to the ONUs in the SPS of the current cycle. Notice that in this case the EPON link may remain idle during the walk time of the Gate messages, unlike the original IPACT scheme which achieves a high degree of interleaved polling.

Secondly, one of the goals of the DPOA scheme appears to be the reduction of delay of the ONUs in the SPS of a scheduling cycle. It is unclear as to how this delay is calculated except that the paper mentions that the delay is averaged across the entire EPON system. Consider the *mean EPON delay per bit* ($\bar{D}_b$) for all the traffic carried by the EPON link in a single cycle defined as

$$\bar{D}_b = \frac{1}{B_{cycle}} \sum_{i=1}^{B_{cycle}} d(i), \tag{3.25}$$

where $d(i)$ is the time from the beginning of the grant, at which bit i is scheduled to arrive at the OLT and B_{cycle} is the total number of bits transmitted in a single scheduling cycle of the DPOA scheme. Assuming walk times can be perfectly interleaved with transmissions, the mean EPON delay per bit will be invariant under any permutation of ONU polling order. Thus, as far as mean EPON delay per bit is

concerned, the polling order makes no difference. Next, Consider the *mean delay per bit per ONU* ($\bar{D}_{b,ONU}$), i.e., the average of the average delay per bit for all the ONUs defined as

$$\bar{D}_{b,ONU} = \frac{1}{N}\sum_{j=1}^{N}\frac{1}{q(j)}\sum_{i=1}^{q(j)} d_j(i), \tag{3.26}$$

where $d_j(i)$ is the time from the beginning of the grant of ONU j, at which bit i is scheduled to arrive at the OLT, $q(j)$ is the length of the traffic transmitted by ONU j in the cycle and N is the total number of ONUs in the EPON. It can be shown that this mean delay per bit per ONU is minimized when ONUs are served in the *increasing* order of queue lengths. (This follows from results for Shortest Job First scheduling where the completion time is to be minimized [92, 96].) Thus, assuming efficient interleaving of polling messages, the DPOA scheme *does not* optimize the mean delay per bit per ONU. Similarly, the maximum EPON delay and maximum delay per bit per ONU can also be defined in a natural way and compared. It is unclear which delay is discussed by the original paper.

Finally, consider urgency of a packet k from real-time queue j at time t, $U_k^j(t)$ as defined by the PIS scheme and described in Eqn. (3.21). Let the *adaptive* cycle time of the IPACT with DPOA and PIS scheme be $T_c > 0$. Suppose ONU i receives a grant of size $0 < g^i < T_c$ starting at time $t_g \geq 0$. Let $0 < L_{max}^{nr} \leq g^i$ be the maximum size of any non real-time queue packet. Consider a packet k in the real-time queue j of ONU i with urgency:

$$U_k^j(t_g) = g^i + L_{max}^{nr} < T_c. \tag{3.27}$$

(Clearly, such a value of urgency $U_k^j(t_g)$ is possible by choosing appropriate values for d_k, L_k and the arrival time of the packet k.) Then, for every such packet, $\forall\, 0 \leq \Delta \leq g^i$:

$$\begin{aligned} U_k^j(t_g + \Delta) &= U_k^j(t_g) - \Delta \\ &\geq L_{max}^{nr}. \end{aligned} \tag{3.28}$$

Hence, as per rule (2b) of the PIS scheme, packets from the non real-time queue will be transmitted instead. However, at time $t = t_g + g^i + L^{nr}_{max}$:

$$\begin{aligned} U^j_k(t_g + g^i + L^{nr}_{max}) &= U^j_k(t_g) - g^i - L^{nr}_{max} \\ &= 0. \end{aligned} \tag{3.29}$$

But, since $g^i + L^{nr}_{max} < T_c$, therefore, $t_g + g^i + L^{nr}_{max} < t_g + T_c$. Hence, at $t = t_g + T_c$, $U^j_k(t) = -(t_g + T_c) < 0$. Thus, the delay bound of packet k from the real-time, high priority queue j of ONU i will be guaranteed to be violated when ONU i receives its next grant in the next cycle. Whereas packet k could have been transmitted during the current grant, it was delayed only in accordance with the PIS rules. Therefore, clearly, the PIS rules must take the cycle time T_c into account when calculating the urgency for any packet k. A further issue is that with schemes such as IPACT which have variable cycle time, the value of T_c may not be fixed or known and may have to be estimated by the ONU. In either case, a more cautious rule for PIS, in addition to the rule (2b) would be:

> If current time is t and $t_g + g^i - t - L^{nr}_0 < L_k$, i.e., if transmitting the non real-time packet would leave a grant too small to transmit the real-time packet, and if $U^j_0(t + t_g + T_c) \leq 0$, i.e., if the delay bound of the real-time packet would be violated by the time of the next grant, then transmit the real-time packet, else transmit the non real-time packet.

The simulation results do not report the throughput in case of the PIS scheme. Specifically, the throughput for the real-time queue would show the amount of real-time traffic affected by the weaker PIS rule.

3.9 Zhu et al.: Urgency Fair Queuing [122]

A new paper by Zhu, Ma et al. [122] continues the work discussed in the previous section [80]. In this paper, the previous algorithm is modified slightly to accommodate the issues pointed out in the previous section. In the new incarnation of the algorithm, known as Urgency Fair Queuing (UFQ), a best-effort queue packet is transmitted only if each of the packets at the head of the real-time queues j have urgency values

$$U_1^j(t) \geq T_c + G, \tag{3.30}$$

where G is defined as a guard time (not to be confused with the transmission guard band defined by the IEEE standard at the physical layer). Thus, if a real-time queue packet cannot wait for the next cycle, then a best-effort queue packet will not be transmitted in its place. This partially addresses the issue pointed out in the previous section. However, other issues remain unaddressed. The paper claims that the new algorithm provides better service to best-effort traffic without violating the QoS requirements of real-time traffic. This can be true only if certain other assumptions about the traffic are granted. For example, suppose that the urgency value of the packet at the head of this queue is larger than T_c+G. Then, in accordance with the UFQ rules, packets from the best-effort queue will be transmitted in its place. Notice that the condition 3.30 only tests the packet at the head of the queue. Packets following the head packet could have urgency values arbitrarily close to that of the head packet. Thus, a scenario is possible where although the head packet (and hence all the other packets) can be deferred to the next cycle, the total size of these packets taken together may be so large that they may not all fit in the grant in the next cycle. Some of them may thus have to be further deferred and may eventually violate their delay bounds. Clearly, this scenario occurs only as a result of the UFQ rules. If some real-time queue packets had been transmitted in the earlier grant despite their lax urgency requirements, then the remaining packets could have all been accommodated

in the next grant. Also note that the scenario we concoct is practical and may even be frequent if one considers the burstiness of traffic: if packets arrive in bursts, then their urgency values could well be close to each other. Thus, contrary to its claim, in its current form, the UFQ algorithm does violate the QoS requirements of the real-time queue in its effort to provide better service to best effort traffic.

The above discussion suggests that the UFQ scheme suffers from the following problem: either it is not sufficient to check the urgency value of only the head packet of a real-time queue, or it is not sufficient to just test the urgency value against the cycle time.

Moreover, the new algorithm eliminates the threshold rule (1) present in the previous version of the algorithm [80]. In many cases, the problems mentioned above could be avoided if the threshold were chosen carefully so as to account for the grant size and maximum total size of packets queued for transmission. However, neither the current paper [122] nor the earlier paper [80] provide any method for estimating these thresholds (Q_H and G).

3.10 Kamal et al.: Prioritized MPCP [63]

The approach of Kamal et al. [63] to the design of a DBA algorithm is quite different from the ones we have seen so far. We describe their scheme briefly below and refer to the reader to the excellent description in the original paper.

3.10.1 Description

Kamal et al. consider the case of N ONUs each with P different priority queues. Their design is motivated by the desire to provide a relatively lower and bounded delay to a higher priority queue compared to a lower priority queue. They claim that the scheme proposed in the paper achieves this goal. In their proposal, called

Prioritized MPCP, grants intended for a higher prirority queue are delivered earlier than those to a lower priority queue, whenever such a choice is possible. Firstly, in Prioritized MPCP, a Gate message containing a grant for low priority traffic scheduled to begin at some time t in the future is only sent out to the destined ONU i with an RTT $2d_i$ at time $t - 2d_i$, i.e., at the latest possible time. Thus, although the decision to allocate a grant to some ONU i at some time t may be made at time $t' \leq t - 2d_i$, the actual grant is not sent out and hence the transmission time allocated by the grant not irreversibly "committed" until time $t - 2d_j$. The grant is tentatively allocated and the Gate message queued for transmission during the interval $[t', t - 2d_i)$. Now suppose a low priority grant has been so allocated at time t (to be sent out at time $t' - 2d_i$). If a new Report for high priority traffic is received at any time in the interval $[t', t - 2d_i)$, then the Prioritized MPCP scheme can revoke its earlier scheduling decision and defer the lower priority grant to a later time, i.e., the lower priority grant can be postponed in favor of a grant for the higher priority traffic. A grant can now be allocated to the higher priority traffic earlier and the delay that the high priority traffic would have faced by waiting for the lower priority grant to finish is eliminated. Since high priority traffic from any ONU can preempt lower priority grants to itself as well as any other ONU, high priority traffic may achieve lower delay with this scheme than with other schemes that perform intra-ONU scheduling at each individual ONU. (Such schemes can insulate the high priority traffic at an ONU only from the lower priority traffic originating at the same ONU.) Fig. 3-4 reproduces (and corrects) the example from the original paper.

3.10.2 Discussion

Prioritized MPCP is proposed as a scheme which would provide lower delay to high priority traffic as compared to lower priority traffic. In [63], the authors claim, uncon-

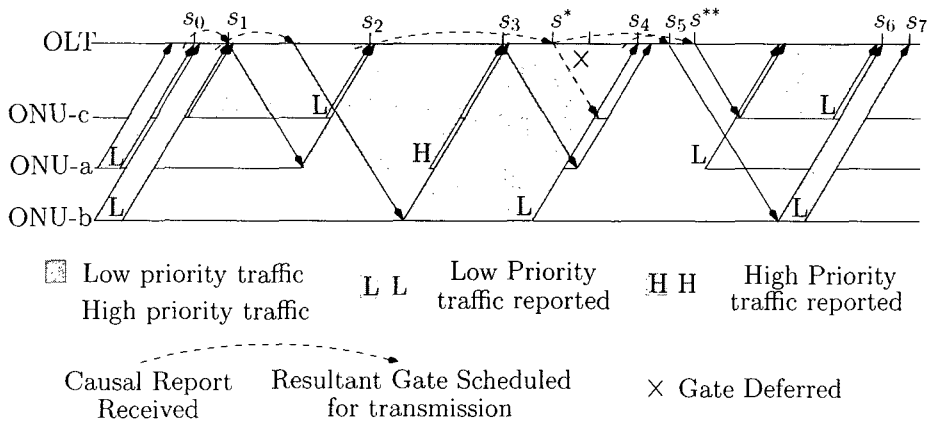


Figure 3-4: An example of the Prioritized MPCP scheme by Kamal et al.

ditionally, that the proposed scheme successfully provides this guarantee. However, this may not be true under arbitrary conditions. The proposed scheme derives its ability to make preferential scheduling decisions favoring higher priority traffic from the flexibility of the deadlines for transmission of Gate messages containing lower priority grants. However, the constraints implied by a relatively wide array of incipient loads and the RTTs could very well eliminate this flexibility, thus disallowing the scheme from deferring Gate transmissions. Without the power to defer low priority grants, the proposed scheme would be severely handicapped and would not be able to provide preferential treatment to higher priority traffic. More precisely, neglecting the time to transmit a Gate message and process it, under the condition that load

$$\lambda \leq \frac{1}{N}\left(\frac{1}{\delta} - \frac{r(N-1)}{2d}\right), \tag{3.31}$$

the proposed scheme would be required to send every Gate message without any further delay. In terms of the variables used in the original paper, the above condition ensures that always $S^j = clock + RTT_i$, giving $X_k^j = clock$ since the ratio of the load to the RTT is low. Thus, the Prioritized MPCP scheme provides the claimed delay differentiation only under a subset of the possible load and RTT values outside of which the scheme behaves similar to the ordinary MPCP protocol with no priority

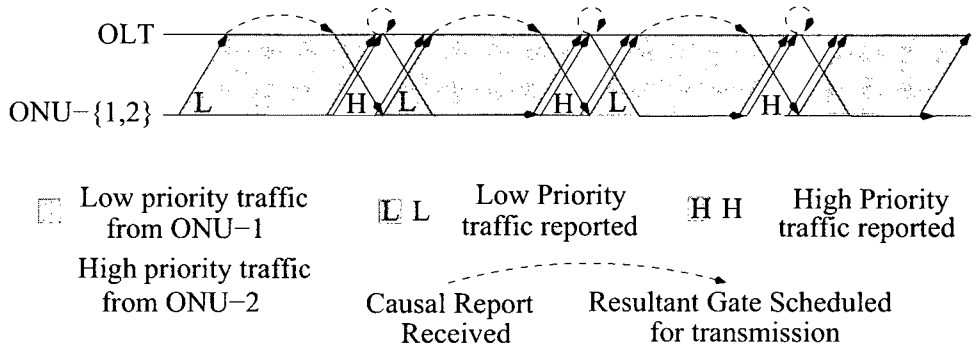


Figure 3-5: A scenario where the P-MPCP scheme does not provide lower delay to high priority traffic.

provisions.

A more subtle issue for the proposed scheme is that of fairness. Traditionally, the issue of fairness has been discussed in the context of fair allocation of grant sizes. For the Prioritized MPCP scheme, the issue of fairness arises in the context of the delay provided to ONUs at different RTTs. Consider two ONUs with one at a much smaller RTT than the other, each with two priority classes $\{i, j\}, i < j$. Observe that according to the definition of the proposed scheme, the time by which the transmission of a low priority Gate message to any ONU can be delayed is inversely related to the RTT to that ONU. Thus, in our example, any low priority Gate message to the nearer ONU would always be deferred for much longer than any low priority Gate message

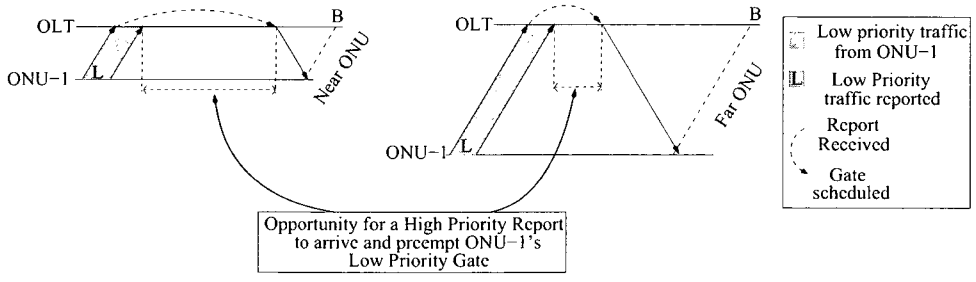


Figure 3-6: A scenario showing the unfairness in the P-MPCP scheme to nearer ONUs.

to the farther ONU. Consequently, a high priority Gate message to the farther ONU has a better chance of being given priority over a low priority Gate message to the nearer ONU. On the other hand, a high priority Gate message to the nearer ONU has a lower chance, (even if only slightly), of being given priority over a low priority Gate message to the farther ONU since such a message would be deferred for a much smaller period of time. While the benefits and deficits of RTT may become reversed if the low and high priority Gate messages under comparison are to the same ONU, by carefully choosing the RTTs, the argument above can be extended to more than two ONUs to arrive at the same conclusion: the Prioritized scheme may penalize nearer ONUs unfairly with a larger delay compared to farther ONUs. Notwithstanding, the effect of such unfairness will diminish as the offered load increases to high values (certainly much higher than those prescribed by condition (3.31)) compared to the RTT, since the RTT is bounded by the IEEE standard at a maximum of 200 μs.

Overall, the Prioritized MPCP scheme proposed by Kamal et al. offers a novel approach to the design of DBA algorithms much different from competing schemes. The idea of deferring scheduling decisions, although not new, has been an important idea in the design of scheduling algorithms. Thus, the design of the Prioritized MPCP scheme represents a correct step towards designing optimal algorithms for DBA in EPONs.

3.11 Kramer et al.: IPACT [71, 72, 70]

3.11.1 Description

The Interleaved Polling with Adaptive Cycle time (IPACT) was one of the earliest schemes proposed for bandwidth allocation in the EPON. The IPACT scheme follows from the simple observation that the OLT need not wait for previous and scheduled grants to complete before allocating future grants. Fig. 3-7 illustrates the typical

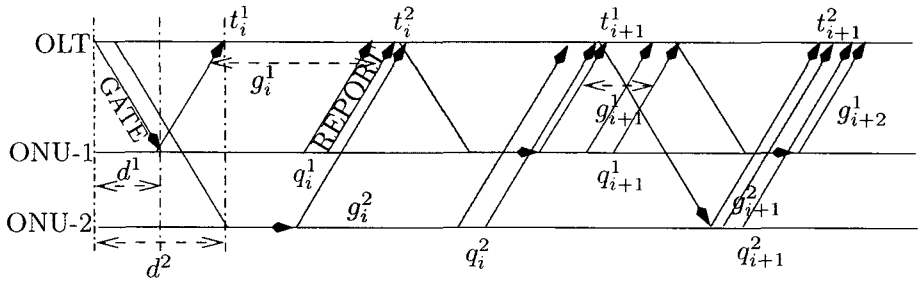


Figure 3-7: An illustration of the IPACT algorithm.

behavior of the IPACT algorithm. We discuss IPACT in detail in [24] and also refer the reader to [71, 72, 70][68].

3.12 Byun et al.: Control-theoretic Grant Size Estimator for IPACT [26]

Byun et al. [26] propose a simple extension to the basic IPACT scheme. Their proposal is motivated by the following problem inherent to the EPON architecture. There exists a significant delay between the measurement of the queue length at the ONU and the reception of this information at the OLT. A corresponding delay also exists between the subsequent computation of the grant size (based on the somewhat "stale" queue length measurement) at the OLT and the beginning of the allocated grant at the ONU. Consequently, it is likely that the size of the grant actually allocated to an ONU is significantly smaller than the actual length of the queue at the ONU at the time of the allocated grant. Thus, a naive DBA scheme that uses the reported queue length information in grant size computation without taking into account the staleness of the information will very likely underestimate the appropriate size of the grant to be allocated to an ONU.

3.12.1 Description

Byun et al. propose a simple yet ostensibly effective estimation heuristic for the calculation of the appropriate grant size. In their scheme, an ONU (essentially) reports the error $e(i,j)$ in OLT's estimation of the queue length at ONU j in any cycle i. If the OLT has overestimated the queue length, then $e(i,j) > 0$, else $e(i,j) < 0$. Using this feedback, the OLT adjusts its estimate of the grant size $g(i+1,j)$ for ONU j for the next cycle $i+1$ as:

$$g(i+1,j) = g(i,j) - \alpha \cdot e(i,j). \tag{3.32}$$

Thus, a positive $e(i,j)$ will lead to a reduction in $g(i+1,j)$ whereas a negative $e(i,j)$ will result in an increase in $g(i+1,j)$ guided by the positive weight *alpha*. Further, using arguments from control theory, Byun et al. show that $0 < \alpha < 2$ provides a stable control algorithm resulting in a steady state error $e(i,j) = 0$, i.e., the OLT will be able to estimate the queue length with high accuracy. They therefore conclude that if the queue length in the steady state is zero, then packet delay and loss is minimized.

3.12.2 Discussion

An attractive feature of the proposed estimation algorithm is its simplicity. From preliminary results reported in [26], the scheme appears to outdo the original IPACT scheme with lower delay and smaller queue length at the ONU. While the scheme, in its current form, does not provide differentiated QoS to ONUs, it seems relatively straightforward to integrate it into a scheme that does (e.g., LAER+EA [15] or HG [102]). It is important to note that while the simulation results reported in the [26] are indeed based on a realistic traffic model (self-similar with long range dependence), the arguments from control theory hold only when the traffic is "piecewise constant with jumps occurring seldom" [26].

3.13 Yang et al.: Delta Dynamic Burst Polling [116]

Yang et al. propose a dynamic bandwidth allocation scheme for IEEE 802.3ah standard based Ethernet Passive Optical Networks. The paper proposes three related theses: 1) Bandwidth allocation to an ONU i should be a function of not only the queue size reported by ONU i but also that reported by all the other ONUs 1a) To be able to use this information, all bandwidth grant (GATE) messages must be sent only after queue reports from all ONUs have been received 2) To support differentiated QoS for different traffic classes, classes must be served in order of their priority and in proportion to their assigned weight and 3) Traffic estimation should be used to predict the amount of arriving traffic in order to provide better delay performance. The paper proposes an algorithm that realizes these goals and evaluates its performance via simulation. The results are compared with two other schemes: naive fixed slot allocation (FSA) and DBA1, a scheme proposed in one of the references. The results demonstrate the superior performance of the proposed scheme.

3.14 Other DBA algorithms

Since the survey by McGarry et al. [83], many new algorithms have been proposed to address the DBA problem in EPONs. The IPACT [71][72][70] algorithm designed by Kramer et al. is one of the earliest proposed algorithms for bandwidth allocation in EPONs, and the main idea can be easily extended to other access network architectures. The simplicity of the IPACT algorithm makes it an attractive solution for dynamic bandwidth allocation in next-generation access networks as well. For example, recently, a "WDM" version of the IPACT algorithm has been proposed as a bandwidth allocation algorithm for WDM-EPONs [84][82]. However, it is important to note that IPACT belongs to the class of DBA algorithms which use temporally local information. Using results from scheduling theory, it may be possible to bound from

below the delay performance obtainable from any such algorithm in the best case [23]. Exploration of the performance limits of this kind for access network architectures and algorithms is a primary objective of our work.

Since LAER suffers from unfairness and overgranting, recent proposals by Shami et al. [103][16][17] have tried to address some of these issues using a more robust definition of weighted max-min fairness. (However, the question of overgranting remains only partially addressed.) New work by Jiang et al. proposes new ways to calculate relative priorities (weights) [62]. Son et al. [108] also provide a max-min fair DBA algorithm. Cyclic Polling [59] is another example of a frame-based DBA scheme. Many other algorithms exist in literature; we omit their details for brevity: adaptive fairness algorithm by Dhaini et al. [39], a class-of-service DBA scheme by Naser et al. [85] and a hierarchical scheme by Zhu et al. [123].

Some algorithms also offer solutions to the problem of stale information due to the large propagation delays involved. For example, Luo et al. [79] propose a prediction method for estimating the amount of new traffic accumulated at the ONUs. Yang et al. [116] simply use the first-order difference. Many algorithm designs view the DBA problem as comprising an inter-ONU and an intra-ONU scheduling problem. Ghani et al. [45][44] and Chen et al. [32] propose improved intra-ONU schedulers for the DBA problem. Kramer et al. [69] propose a novel inter-ONU scheduler which guarantees sibling as well as cousin-fairness. Their approach is based on an approximation of the actual aggregated demand of subscriber queues. Zhang et al. propose a GPS-fair scheduler for EPONs [117]. Naser et al. propose a joint-interval scheduling scheme [86]. Architectures different from the IEEE 802.3ah EPON have also been proposed. For example, An et al. [10] and Foh et al. [43] propose a different architecture for EPONs.

Chapter 4

Expected grant size of the gated IPACT scheme for EPONs

4.1 Introduction

Interleaved Polling with Adaptive Cycle Time (IPACT) is one of the earliest proposed polling schemes for dynamic bandwidth allocation in Ethernet Passive Optical Networks (EPONs) and has been extensively used as a benchmark by many subsequent allocation schemes. In this chapter, we attempt to construct a mathematical model of the IPACT scheme under the gated service discipline. For $N = 1$ ONU, we derive a closed-form expression for the steady state grant size. For $N > 1$ ONUs, we need to consider separately a small and a large load-distance ratio. For the former case, the $N = 1$ ONU model holds even for $N > 1$. For the latter case, we find a closed form expression for the grant size. Our model shows a reasonable match with the values obtained from simulation for the steady state queue size and hence the throughput and delay.

An *Ethernet Passive Optical Network (EPON)* is a point-to-multipoint, bidirectional, high rate optical network for data communication. The EPON link is shared by multiple users. Each user connects to the EPON link through a device known as an *Optical Network Unit (ONU)*. Since the link is shared, link use must be centrally arbitrated. This function is performed by a single special device called the *Optical*

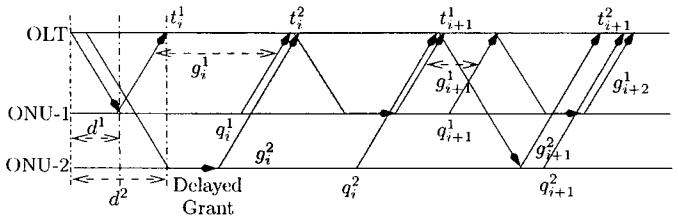


Figure 4-1: An example with two ONUs illustrating notation used for the recursive model.

Line Terminator (OLT). The direction of communication from the ONUs to the OLT is known as *upstream* direction whereas the direction from the OLT to the ONUs is known as the *downstream* direction. The data rate in each direction is set to 1 Gbps by the IEEE EPON standard [3]. Overall, the link exhibits a tree topology with the OLT at the root of the tree and the ONUs at the leaves. The EPON link is shared by all users in the upstream direction. The OLT decides which ONU is allowed to transmit data and for how many bytes. The OLT uses a special control message called a *Gate* to grant transmission opportunities to ONUs. Appended to the data traffic, the ONU also transmits a control message containing a *Report* of the number of bytes buffered in its queue, waiting for a subsequent transmission opportunity. An algorithm implemented in the OLT, which uses these reports and gate messages to construct a transmission schedule is known as a dynamic bandwidth allocation (DBA) algorithm.

4.2 The IPACT Protocol

Interleaved Polling Scheme with Adaptive Cycle Time (IPACT) is a DBA scheme for EPON proposed by Kramer et al. [71][72]. IPACT is one of the earliest dynamic bandwidth allocation schemes for EPONs and has been extensively used as a benchmark by many subsequent allocation schemes [81][15][26][70][43][79][115]. To our knowl-

edge, this is the first attempt to provide an analytical model for the IPACT scheme. The evaluation in [68] is based on simulations and focuses on service disciplines.

IPACT is an algorithm for interleaved polling of ONUs designed to minimize the walk times. For example, if the OLT sends a grant message to an ONU and then waits for the ONU to send data before sending a grant message to the next ONU, then, the waiting will result in wastage of a significant amount of data bandwidth thus decreasing link utilization. IPACT suggests interleaving the polling messages to consecutive ONUs in order of decreasing distance from the OLT such that transmissions arrive at the OLT as tightly packed as possible. As illustrated in Fig. 4-1 the OLT has sufficient information to schedule the transmissions for ONU-2 before the data from ONU-1 arrives at the OLT. This is because the OLT knows the distance to each ONU and also knows the size of the grant it allocated to each ONU. Hence the OLT can calculate the time of the arrival of the last bit from each ONU and can therefore schedule the transmission from the next ONU right after the one from the previous ONU has terminated. From Fig. 4-1 for example, at time t_i^1, the OLT knows the time at which the transmission from ONU-1 will end and can therefore send the Gate message to ONU-2 to schedule its transmission to arrive at the OLT at time t_i^2. In this way, the interleaving helps minimize link underutilization during walk times. However, this is not always possible depending on the distance of the ONU from the OLT. To address this issue, the original IPACT scheme also includes other optimizations as well as procedures for detecting arriving and departing ONUs. However, we do not include these portions in our analysis. We focus on the core IPACT bandwidth allocation algorithm which is a simple client-server protocol with quite general applicability.

Another parameter of the IPACT algorithm is the allocation policy of the "server", i.e., the OLT. When the OLT receives a report of the queue length from an ONU, it

need not grant a single transmission slot in the current cycle for the buffered contents in their entirety. Instead, the OLT may fix a particular policy such as granting a time slot: a) of fixed length regardless of the reported queue length (static allocation), b) equal to the reported buffer length but bounded by a maximum (limited allocation), c) larger than the reported queue length in anticipation of future traffic (credit-based allocation) or d) equal to the reported queue length (gated allocation). Other variants are also possible. In this work, for simplicity, we focus on the the gated allocation discipline. In the following sections, we analyze this scheme using a recursive model.

4.3 Background and Related Work

Polling systems have been researched widely for a number of years [111][105][34]. A general analysis of a polling system seems to be a challenging problem and the results available are a product of complex analyses and therefore involve many simplifying assumptions and approximations. The analysis is further complicated if one wishes to include a realistic traffic model. In this work, we attempt to derive a model of the IPACT scheme from its basic definition. Such a clean-slate analysis can provide clearer insight into the dynamics of the scheme with a minimal set of assumptions and simplifications. Our main goal is to provide a clear and simple yet detailed model of the IPACT scheme derived from its definition. In the end, we hope to obtain simple closed-form expressions relating the grant size (and hence the delay and utilization) to other parameters such as the load and round-trip time. We believe that our work can provide useful guidelines in the design of new bandwidth allocation schemes—currently an area of intense research [81][15][26][70][43][79][115]. Recently, Park et al. [91] obtained new results about the performance of the IPACT scheme under the gated allocation policy. Their approach is novel and comprehensive and provides strong results. However, their assumptions and results are different and their method,

more sophisticated and complex.

4.4 A Recursive Model for IPACT

An accurate model for the IPACT scheme must account for the recursive relationship between the transmission times, queue lengths and the grant sizes in successive scheduling cycles. In this section, we develop such a model. We will use the notation as defined in Table 4.1 and illustrated in Fig. 4-1.

Table 4.1: Definition of notation used

Notation	Definition	Units (value)
t_i^j	The time of arrival of the i^{th} transmission by ONU-j at the OLT	seconds
q_i^j	The queue length at ONU-j after completion of the i^{th} data transmission	bits
g_i^j	The size of the grant allocated by OLT to ONU-j for its i^{th} transmission	bits
d^j	$\frac{1}{2}$RTT to ONU-j	seconds
λ	The average arrival bit-rate	bits/second
δ	The time to transmit one bit over the EPON	seconds/bit (10^{-9})
r	The size of the Report message	bits (512)
b	The size of the guard band	seconds (2×10^{-6})
m	The size of the Gate message	bits (512)
N	Total number of ONUs in the EPON	—

Our goal is to express queue lengths, grant sizes and transmission times in terms of parameters such as the input traffic rate (λ), link rate (δ^{-1}), number of ONUs (N), and the distances of individual ONUs (d^j) from the OLT.

4.4.1 Calculation of grant size g_i^j

The size of the grant issued by the OLT to an ONU is based on the queue length of the ONU. The ONU informs the OLT about the length of its queue in a Report message. However, the ONU must be granted a transmission slot to transmit the Report message as well. As per the IPACT protocol, the ONU always appends a Report message to each transmission. Let r denote the length of a Report message specified by the IEEE EPON standard [3]. The OLT receives information about the current size of the queue length of an ONU only at the end of a current transmission by that ONU. The OLT can use this information only to decide the size of the next transmission slot to be granted to that ONU. Under the gated service discipline, the OLT always allocates exhaustive grants, i.e., the transmission slots are always large enough to transmit all the data in the ONU queue reported to the OLT. Then, for gated service, the size of a grant is equal to the queue size reported in the previous grant, with an additional r bits allocated for the next Report message. Thus,

$$g_i^j = q_{i-1}^j + r, \quad i > 1. \tag{4.1}$$

What happens when an ONU is granted its first transmission opportunity? The OLT has no information about the queue size of the ONU, since the ONU has not transmitted any Report messages to the OLT at all. In this case, we assume that the OLT grants the ONU a transmission slot of a minimum size large enough to transmit a Report message. Thus, for any ONU, the size of its first transmission opportunity will be

$$g_1^j = r. \tag{4.2}$$

4.4.2 Calculation of initial queue length q_1^j

Next, we calculate the initial queue length at any ONU, as required by (4.1). Recall that for any ONU-j, the first transmission comprises only a Report message and none of the buffered data as per Eqn. 4.2. Traffic arrives at the rate of λ bits/second. Suppose an ONU-j is d^j seconds away from the OLT. Suppose that the first bit of the ONU's first transmission reaches the OLT at time t_1^j. Since ONU-j is d^j seconds away from the OLT, it must have transmitted the first bit of its first transmission at time $t_1^j - d^j$. The amount of traffic accumulated at ONU-j during this interval will therefore be $\lambda(t_1^j - d^j)$ which will be the queue length reported in the first Report message by ONU-j. Thus,

$$q_1^j = \lambda \cdot (t_1^j - d^j). \tag{4.3}$$

The first Gate message from the OLT to poll the first ONU will reach ONU-1 at time d^1. Therefore, the first Report message from ONU-1 will reach the OLT at time $t_1^1 = 2d^1 + \delta m$. Thus,

$$q_1^1 = \lambda(d^1 + \delta m). \tag{4.4}$$

For the first cycle, i.e., all transmissions t_1^j, the length of each transmission will be equal to the length of the Report message. Thus, an ONU-j, $j > 1$, will have to wait at least δr after t_1^{j-1} before transmitting its own Report message. On the other hand, an ONU-j cannot transmit its Report message until it has been polled by the OLT. This takes time at least d^j. Hence, for any ONU $1 < j \leq N$, the time at which its Report message arrives at the OLT will be

$$t_1^j = \max(t_1^{j-1} + \delta r + b, j\delta m + 2d^j). \tag{4.5}$$

4.4.3 Calculation of queue length q_i^j after the i^{th} transmission ($i > 1$)

The queue length q_i^j at the end of i^{th} transmission by device j is exactly equal to the new traffic arrived since the last queue length measurement. To find this queue

length, we need to find out the times at which queue lengths are measured and then using the arrival rate λ obtain the queue length. From Fig. 4-1 we observe that q_{i-1}^j is the length of the queue at the time the Report message is transmitted at the ONU. Since any transmission by the ONU takes time d^j to reach the OLT and since the first bit of the i^{th} transmission reaches the OLT at time t_i^j, the first bit must be transmitted at the ONU at time $t_i^j - d^j$. Further, since the grant size is g_i^j, it must take δg_i^j time for the transmission to complete. Thus, the time at which the ONU sends its last bit must be $t_i^j - d^j + \delta g_i^j$. The queue length is measured δr time before this time at the ONU. Thus, the time at which the queue length is measured after the i^{th} transmission by device j is

$$t_i^j - d^j + \delta g_i^j - \delta r.$$

Thus, the queue length after the previous i.e., $i-1^{th}$ transmission would be measured at time

$$t_{i-1}^j - d^j + \delta g_{i-1}^j - \delta r.$$

Therefore, the amount of new traffic accumulated since the last queue measurement upto the current queue measurement will be

$$\begin{aligned} q_i^j &= \lambda \cdot \left[(t_i^j - d^j + \delta g_i^j - \delta r) - (t_{i-1}^j - d^j + \delta g_{i-1}^j - \delta r)\right] \\ &= \lambda \cdot \left[(t_i^j - t_{i-1}^j) + \delta \cdot (g_i^j - g_{i-1}^j)\right], \quad i > 1 \end{aligned} \tag{4.6}$$

4.4.4 Calculation of transmission times t_i^j

Next, we calculate the transmission times for any ONU. If the EPON consists of $N = 1$ ONU, then the transmission time t_i^j depends only on the transmissions in the previous grants. A new transmission can only begin after the previous one has finished and the new Gate message sent out by the OLT reaches the ONU. A message from the OLT takes d^j time to reach an ONU-j. The new transmission from ONU-j

will in turn take time d^j to reach the OLT. Thus, consecutive transmissions from the ONU are separated by $2d^j$ seconds. An ONU's previous transmission beginning at time t_{i-1}^j will end at time $t_{i-1}^j + \delta g_{i-1}^j$. Thus, a new transmission, in the single ONU case will begin at

$$t_i^j = t_{i-1}^j + \delta g_{i-1}^j + 2d^j + \delta m + b, \quad N = 1 \tag{4.7}$$

If $N > 1$, then considering indices modulo N, the transmission from ONU-j cannot begin unless the one from the previous ONU-$(j-1)$ has ended, unless, ONU-j is sufficiently far away from the OLT. In this latter case, the time d^j taken by the Gate message to reach ONU-j sufficiently delays the transmission from ONU-j as illustrated by transmission t_2^2 of ONU-2 in Fig. 4-1. Suppose ONU-$(j-1)$ transmits at time t_i^{j-1}. Its transmission will finish at $t_i^{j-1} + \delta g_i^{j-1}$. No other transmission can begin before this time. Now suppose that ONU-j last transmitted in the previous cycle at time t_{i-1}^j. Then, d^j must be large enough so that ONU-j's next transmission beginning at t_i^j is sufficiently delayed past the time of completion $t_i^{j-1} + \delta g_i^{j-1}$ of the previous ONU. Thus, we arrive at the following condition for the transmission time:

$$t_i^j = \max\left(t_{i-1}^j + \delta(g_{i-1}^j + m) + 2d^j, t_i^{j-1} + \delta g_i^{j-1}\right) + b \tag{4.8}$$

where device index j is counted modulo $N > 1$. This completes the specification of the recursive model.

4.5 Closed-form Solution of the Recursive Model

In the previous section, we expressed the queue length reported by the ONU in its i^{th} transmission in terms of other parameters and variables of our model. In this section, we attempt to derive closed form expressions for the throughput and response time for an ONU. We divide our derivation into separate cases for $N = 1$ (Sec.4.5.1) and $N > 1$ (Sec.4.5.2) ONU(s).

4.5.1 Single ONU

Consider an EPON with a single ONU. Thus, $N = 1$ in the recursive model developed in Sec. 4.4. From, (4.7) we have

$$t_i^j - t_{i-1}^j = \delta g_{i-1}^j + 2d^j + \delta m + b \tag{4.9}$$

Substituting in (4.6) and simplifying gives,

$$q_i^j = \lambda \cdot (2d^j + \delta g_i^j + \delta m + b), \quad i > 1, N = 1.$$

Substituting (4.1) and simplifying gives,

$$g_{i+1}^j = \lambda \cdot (\delta g_i^j + 2d^j + \delta m + b) + r. \tag{4.10}$$

Solving this recurrence gives the steady state grant size,

$$g_s = \frac{\lambda \cdot (2d^j + \delta m + b) + r}{1 - \lambda\delta}. \tag{4.11}$$

The response time R_{1-ONU} for a single ONU when $\lambda\delta < 1$ will be

$$R_{1-ONU} = \delta m + 2d^j + \delta g_s + b = \frac{2d^j + \delta(m + r) + b}{1 - \lambda\delta} \tag{4.12}$$

Figures 4-5, 4-6, and 4-7 show the match between the steady state grant size predicted by Eqn. 4.11 and that obtained by simulation. We have also verified that the response time predicted by Eqn. 4.12 matches simulations but omit the graph due to space limitations.

4.5.2 Multiple ONUs at an identical distance

For $N > 1$, we also assume that all N ONUs are located at an identical distance

$$d = d^j. \tag{4.13}$$

We consider two cases based on the ratio of the load to the RTT to any ONU.

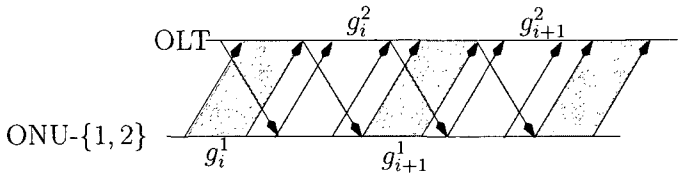


Figure 4-2: The low load-distance ratio

Low Load-Distance ratio

First, consider the case where for $N > 1$ ONUs, the l.h.s. term of the two expressions compared in Eqn. 4.8 is the maximum, i.e.,

$$t_i^{j-1} + \delta g_i^{j-1} < t_{i-1}^j + \delta(g_{i-1}^j + m) + 2d. \tag{4.14}$$

Then by Eqn. 4.8 and Eqn. 4.14:

$$t_i^j - t_{i-1}^j = \delta(g_{i-1}^j + m) + 2d + b, \tag{4.15}$$

and,

$$\begin{aligned} t_i^j &> t_i^{j-1} + \delta g_i^{j-1} + b, \\ &> t_i^{j-2} + \delta g_i^{j-2} + \delta g_i^{j-1} + 2b, \end{aligned}$$

and so on. Continuing, we can write:

$$t_i^j > t_{i-1}^j + \sum_{k=j}^{N} \delta g_{i-1}^k + \sum_{k=1}^{j-1} \delta g_i^k + Nb. \tag{4.16}$$

In other words, using Eqn. 4.14, Eqn. 4.15 and Eqn. 4.16, we can write:

$$\sum_{k=j}^{N} \delta g_{i-1}^k + \sum_{k=1}^{j-1} \delta g_i^k + Nb \leq \Delta t = \delta(g_{i-1}^j + m) + 2d + b. \tag{4.17}$$

where $\Delta t = t_i^j - t_{i-1}^j$ is the cycle time. This corresponds to the example shown in Fig. 4-2 with two ONUs. Clearly, if the grant size to each ONU is smaller than the RTT to the ONU, then neither ONU will ever have to wait for the other ONU. The derivation would proceed after Eqn. 4.15 exactly as for the single ONU case in

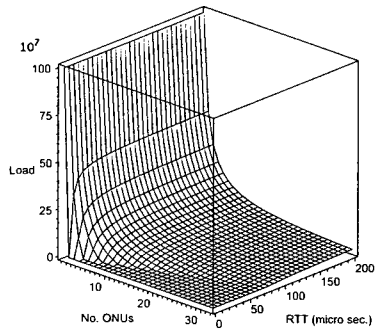


Figure 4-3: Steady state grant size for non-negative load points beneath the surface is given by Eqn. 4.18 and for those above is given by Eqn. 4.26.

the previous section, resulting in the same steady state grant size as arrived at in Eqn. 4.11, i.e.,

$$g^j = g_s = \frac{\lambda \cdot (2d + \delta m + b) + r}{1 - \lambda\delta}. \tag{4.18}$$

Since the steady state grant size g^j is the same for all N ONUs, using Eqn. 4.17 and Eqn. 4.18, we can write:

$$N \cdot (\delta g^j + b) \leq \delta(g^j + m) + 2d + b, \text{ or} \tag{4.19}$$

$$(N-1) \cdot (\delta g^j + b) \leq 2d + \delta m. \tag{4.20}$$

Substituting Eqn. (4.18) shows that the steady state grant size g^j from Eqn. 4.18 holds when:

$$\lambda \leq \frac{1}{N\delta}\left(1 - \frac{(N-1)(\delta r + b)}{2d + \delta m}\right), \tag{4.21}$$

i.e., our assumption Eqn. 4.14 translates into the above condition. In other words, the single-ONU model applies to the multi-ONU case for the load-distance relationship described by Eqn. 4.21. The steady state grant size of any non-negative load point beneath the surface shown in Fig. 4-3 will be given by Eqn. 4.21.

High Load-Distance ratio

Alternatively, consider the r.h.s. term of the two expressions compared in Eqn. 4.8 to be larger, i.e.,

$$t_{i-1}^{j} + \delta(g_{i-1}^{j} + m) + 2d < t_{i}^{j-1} + \delta g_{i}^{j-1}. \tag{4.22}$$

Then by Eqn. 4.8 and Eqn. 4.22:

$$t_{i}^{j} - t_{i-1}^{j} = \sum_{k=j}^{N} \delta g_{i-1}^{k} + \sum_{k=1}^{j-1} \delta g_{i}^{k} + Nb, \tag{4.23}$$

and, by similar reasoning as before,

$$\delta(g_{i-1}^{j} + m) + 2d + b \leq \Delta t = \sum_{k=j}^{N} \delta g_{i-1}^{k} + \sum_{k=1}^{j-1} \delta g_{i}^{k} + Nb. \tag{4.24}$$

If all ONUs are located at the same distance, the cycle time Δt is solely comprised of transmission grants of other ONUs, i.e., those that transmitted after ONU-j in the previous cycle $(i-1)$ and those that will transmit before ONU-j in the current cycle (i). (See the example with two ONUs in Fig. (4-4).) Substituting Eqn. 4.23 into Eqn. 4.6 and simplifying gives

$$q_{i}^{j} = \lambda\delta\left(\sum_{k=j+1}^{N} g_{i-1}^{k} + \sum_{k=1}^{j} g_{i}^{k}\right) + \lambda Nb. \tag{4.25}$$

Assume that a steady state exists (see appendix for details) and that the steady state grant size for each ONU is identical. Then, in steady state,

$$\begin{aligned} g^{j} - r &= \lambda\delta N g^{j} + \lambda Nb \text{ or,} \\ g^{j} &= \frac{\lambda Nb + r}{1 - N\lambda\delta}. \end{aligned} \tag{4.26}$$

Again, from Eqn. 4.24 we know that Eqn. 4.26 holds when

$$\delta(g^{j} + m) + 2d + b < N(\delta g^{j} + b). \tag{4.27}$$

Substituting Eqn. 4.26 and simplifying shows that Eqn. 4.26 holds for the remaining loads

$$\lambda > \frac{1}{\delta N}\left(1 - \frac{(N-1)(\delta r + b)}{2d + \delta m}\right). \tag{4.28}$$

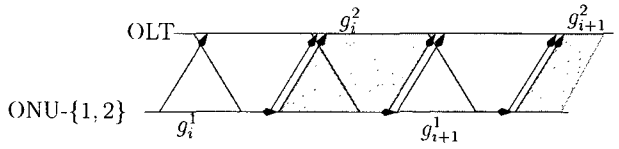


Figure 4-4: High load-distance ratio.

This completes the solution of our recursive model for IPACT under the gated service scheme.

4.6 Simulations and a Comparison

Simulations were conducted in order to validate the above model of the IPACT scheme under a gated service policy. The simulations were conducted for $N \in \{1, 2, 4, 20\}$ ONUs $d \in \{25, 50, 100\}$ μs away from the OLT for loads $0 \leq \lambda \leq N^{-1}$. Each run simulated 10 seconds of real time. For simulations with self-similar traffic, the duration of the simulation must be very large (of the order of hundreds to a few thousands of seconds). Depending on the load to be generated and the number of ONUs, such simulations can be computationally intensive in both time and space. More rigorous simulations of much longer duration are currently in progress. Figures 4-5, 4-6, and 4-7 show the results of the simulation when all ONUs were identically located at distances of 5 km, 10km, and 20 km respectively. The self-similar traffic curve consists of 3000 points, one obtained from each run for a specific load. Fifteen load values were used. Traffic injected into the ONUs was generated by an aggregated source with a measured Hurst parameter of 0.8 [73]. The source generated bursts whose "ON" and "OFF" period lengths followed the Pareto distribution. The integer lengths of the packets in each burst were drawn uniformly at random from the interval [64, 1518] bytes.

Figures 4-5, 4-6, and 4-7 show the match between the steady state queue size

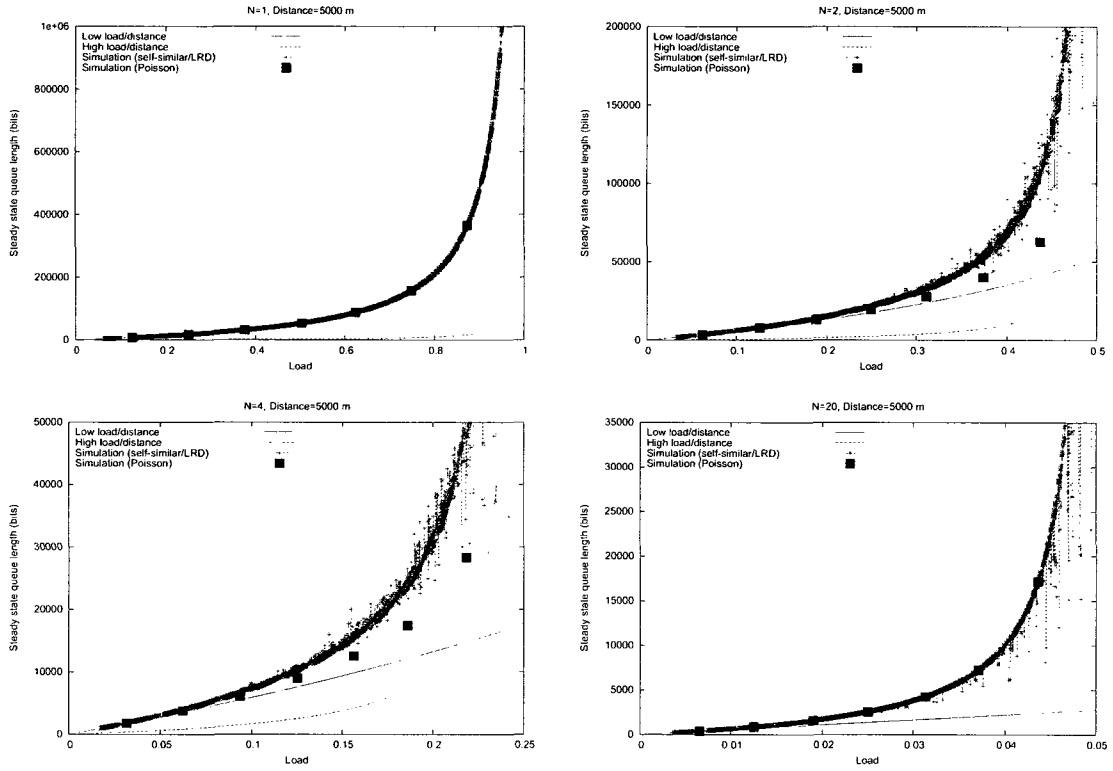


Figure 4-5: Simulation results with Poisson and self-similar traffic for a few selected points from Fig. 4-3 with ONU distance set to 5 km.

predicted by our model and that measured from simulation with both Poisson as well as self-similar/LRD [73] traffic models. The match against the Poisson model is better than that against the self-similar model since our model is based simply on the average load and hence cannot account for the infinite variance exhibited by the bursty self-similar traffic. Nonetheless, the model shows a reasonably and sufficiently good match with both models as far as verifying the correctness of the model is concerned.

4.7 Conclusions and Future work

In this chapter, we attempted to analyze the IPACT protocol for allocation of EPON bandwidth using a recursive formulation. IPACT is one of the earliest and most

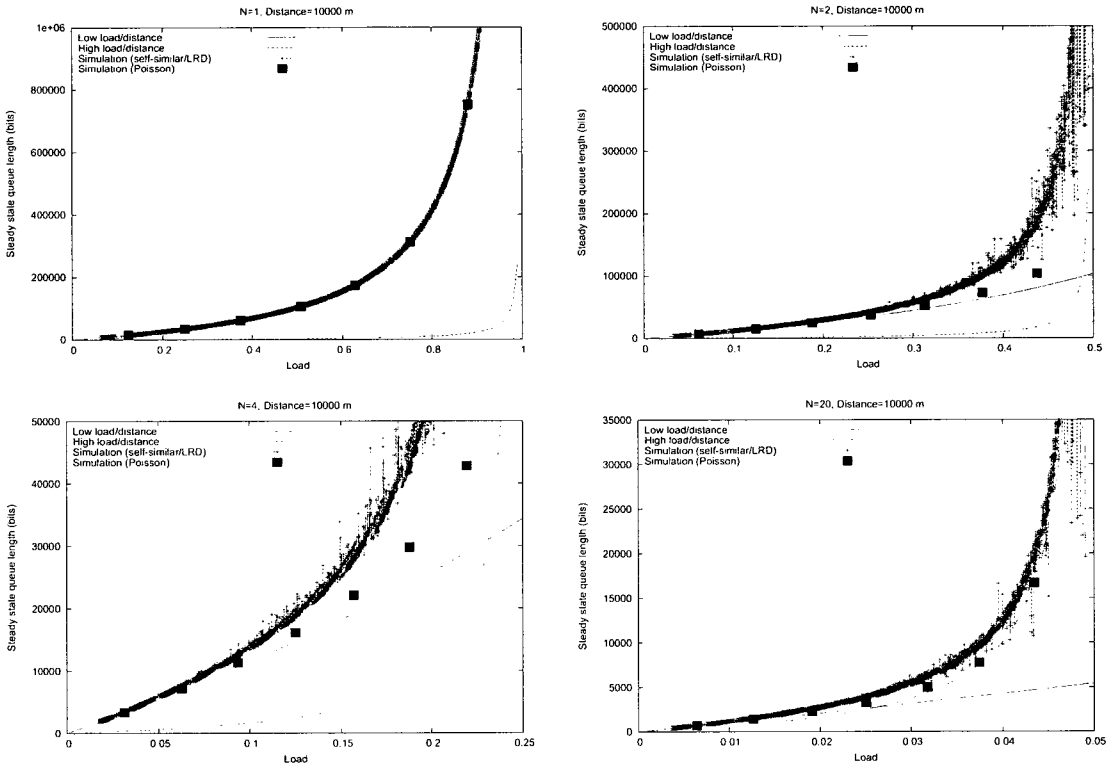


Figure 4-6: Simulation results with Poisson and self-similar traffic for a few selected points from Fig. 4-3 with ONU distance set to 10 km.

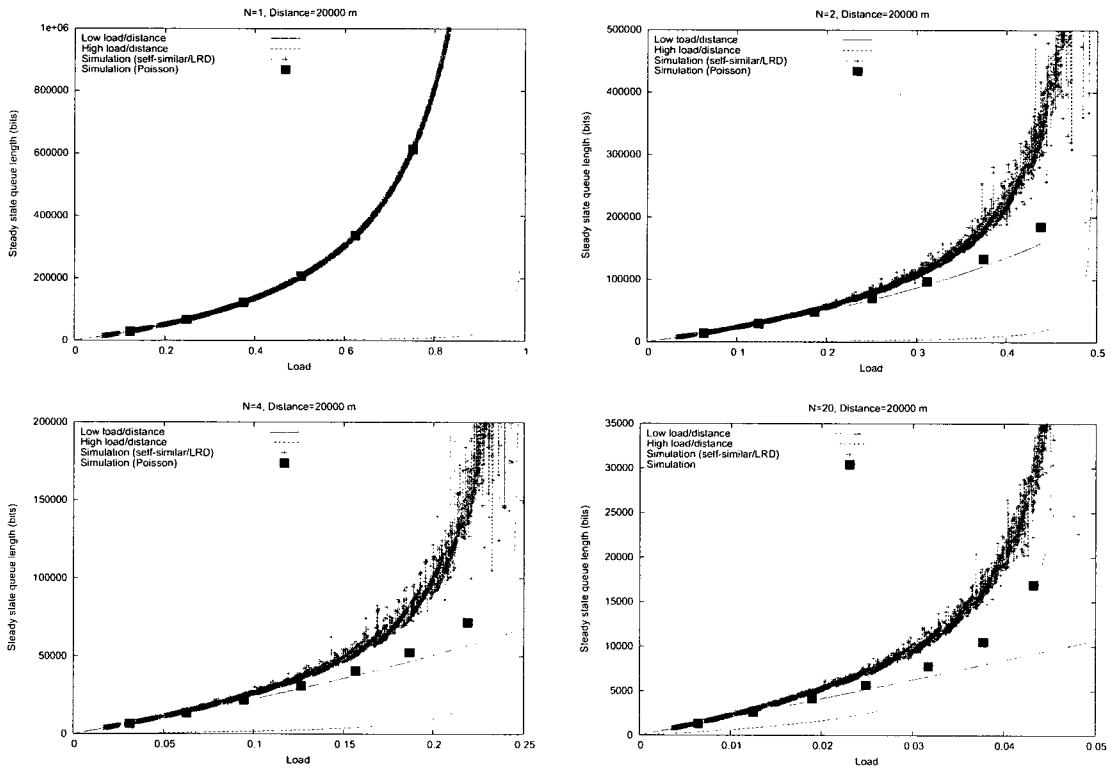


Figure 4-7: Simulation results with Poisson and self-similar traffic for a few selected points from Fig. 4-3 with ONU distance set to 20 km.

widely used benchmarks for bandwidth allocation in EPONs. However, there was no analytical model describing the IPACT scheme. In this chapter, we developed such a simple model. We derived closed-form expressions for the grant size allocated by IPACT under the gated service scheme as a function of the input load, ONU RTT and other protocol parameters.

The main assumption of our analysis is that the fluid traffic arriving in an interval t is λt where λ is the average traffic arrival rate, is known and fixed. However, just as with other approaches, this is a drawback of our analysis as well. For a well-behaved traffic source such as Poisson traffic, a small sample from a small interval t will converge to the actual mean λ with high probability. However, for self-similar, heavy tailed traffic, this does not hold. Since the tail decays only polynomially, a small sample can contain a large deviation from the mean. In such cases, the predictions of the average value from our analysis will exhibit error. Thus, the assumption that the queue reported at the end of a transmission of size t will be λt will introduce considerable error for a heavy tailed traffic source.

4.8 Appendix

Consider Eqn. 4.25. We can write a similar equation for ONU-$j-1$ (using modulo N indexing):

$$q_i^{j-1} = \lambda\delta\left(\sum_{k=j}^{N} g_{i-1}^k + \sum_{k=1}^{j-1} g_i^k\right) + \lambda N b. \tag{4.29}$$

Subtracting, Eqn. 4.29 from Eqn. 4.25 gives:

$$q_i^j = q_i^{j-1} + \lambda\delta(q_{i-1}^j - q_{i-2}^j), \tag{4.30}$$

which can be rewritten as:

$$a_n = a_{n-1} + \lambda\delta(a_{n-N} - a_{n-2N}). \tag{4.31}$$

Equation Eqn. 4.31 can be interpreted as the recurrence describing the IPACT scheme under gated service and high load-distance ratio (as defined by Eqn. 4.28) since it captures the essential behavior of the bandwidth allocation process of the scheme. The characteristic equation of Eqn. 4.31 is

$$p(x) = x^{2N} - x^{2N-1} - cx^N + c = 0, \tag{4.32}$$

where $c = \lambda\delta$. By showing that all the roots of Eqn. 4.32 will be no greater than unity for $c < N^{-1}$, we can argue that a steady state exists in the high load-distance ratio regime as well.

Chapter 5

IPACT with Smallest Available Report First: A New DBA Algorithm for EPON

Dynamic Bandwidth allocation in Ethernet Passive Optical Networks (EPONs) has been an area of intense research in recent years. Most of the proposed solutions offer clever methods for fair grant sizing, traffic prediction, and prioritized, differentiated services. Barring some work by Kamal et al. and some elements in the scheme proposed by Ma et al., no work has been done on exploring the order of granting (i.e., ONU sequencing) in an EPON. In this work, we propose an unexplored heuristic for improving the performance of the IPACT scheme with respect to the most important metric: packet delay. In this heuristic, the OLT always grants that ONU which has the Smallest (Available) Reported queue length, First (SARF). Our simulations indicate that our heuristic can improve the delay performance of IPACT by 10-20% (when tested under the gated allocation policy).

5.1 Introduction

5.1.1 EPON

An *Ethernet Passive Optical Network (EPON)* is a point to multipoint, bidirectional, high rate optical network for data communication. The EPON link is shared by multiple users. Each user connects to the EPON link through a device known as an *Optical Network Unit (ONU)*. Since the link is shared, link use must be centrally arbitrated. This function is performed by a single special device called the *Optical Line Terminator (OLT)*. The direction of communication from the ONUs to the OLT is known as *upstream* direction whereas the direction from the OLT to the ONUs is known as the *downstream* direction. The data rate in each direction is set to 1 Gbps by the IEEE EPON standard [3]. Overall, the link exhibits a star topology with the OLT at the root of the star and the ONUs at the leaves. The EPON link is shared by all users in the upstream direction. The OLT decides which ONU is allowed to transmit data and for how many bytes. The OLT uses a special control message called a *Gate* to grant transmission opportunities to ONUs. Appended to the data traffic, the ONU also transmits a control message containing a *Report* of the number of bytes buffered in its queue, waiting for a subsequent transmission opportunity. The IEEE standard does not specify the actual algorithm to be used for grant allocation and leaves it open for implementation by vendors.

5.1.2 Dynamic Bandwidth Allocation

An algorithm implemented in the OLT, which uses *Report* and Gate messages to construct a transmission schedule and convey it to the ONUs is known as a Dynamic Bandwidth Allocation (DBA) algorithm. DBA in EPONs has been an area of intense research in recent years. Many solutions have been proposed. The challenge in designing a DBA algorithm lies in developing an algorithm that is practical, simple,

efficient and meets service provider requirements. A significant portion of the body of DBA research has focused on fairness in allocating grants [16][15]. The solutions proposed resemble Weighted Fair Queuing in flavor. Another track of research has focused on improving the freshness of the queue information available to the OLT by employing some variant of a traffic prediction algorithm [26][79]. A third track has focused on minimizing the idle periods on the upstream channel by clever interleaving of messaging delays with data transmissions [66][18][102]. The distinction between inter- and intra-ONU scheduling has resulted in new suggested solutions for these two portions [45][44][69].

When it comes to public subscriber access networks such as EPONs, applications such as voice and video are the main sources of revenue for operators. Voice and video are very sensitive to delay and video traffic is quite bursty in nature. For this reason, packet delay is considered to be the most important benchmark to measure the efficacy of any proposed DBA algorithm. There are at least two ways to interpret this benchmark. Much of the existing work appears to focus on bounding the inter-service delay at any ONU, i.e., the time until an ONU is serviced again is guaranteed to be bounded. All frame-based schedulers [15][18][81][60] are based on this approach. The issue of grant sizing is treated as a separate question in this approach. It usually remains unclear as to why the particular chosen cycle or frame length and the particular chosen grant sizing heuristic when taken together would yield a low delay DBA strategy. (In fact, schemes based on Packetized Generalized Processor Sharing-based approach provide fairness to all ONUs only at the cost of delaying all ONUs equally.) In our opinion, a second approach to the DBA problem could focus on minimizing the achievable per packet delay in an EPON. Instead of just bounding the inter-service (also called the "cycle") time per ONU, the following question seems natural and interesting: What is the minimum per-packet delay that can be achieved

under the IEEE EPON architecture? An attempt to provide an answer would likely involve viewing the DBA as some variant of a scheduling problem. Scheduling theory [92], with its rich set of models and results [97], can offer many insights into the basic structure of the DBA problem and in turn shed light on the limits of the performance achievable under the IEEE EPON architecture. Our present work is not devoted to finding a complete answer to the question posed above; the answer may be difficult to arrive at. However, while such a theoretical line of research may be long and arduous, it can provide valuable ideas based on a solid theoretical foundation for novel and highly effective solutions to the DBA problem. In this work, we follow this approach and propose one such solution.

5.2 IPACT with Smallest Available Report First (SARF)

The main contribution of this work is the idea of allocating grants in an order that minimizes packet delay. To this end, we propose the use of the Smallest Available Report First heuristic and through simulations, demonstrate its efficacy in reducing packet delay.

5.2.1 IPACT

Kramer et al. proposed the simple Interleaved Polling with Adaptive Cycle Time scheme as a solution for DBA in EPONs [71][70]. One of their main contributions is the observation that an OLT need not wait for a transmission from an ONU to finish, before sending a Gate message to the next ONU. The IPACT scheme can therefore transmit downstream control messages to the ONUs while receiving data transmissions from other ONUs in the upstream, thus minimizing upstream underutilization due to walk time. This can be a significant problem in public access networks with high bandwidth and delay products. In addition, Kramer et al. also propose various

policies for calculating the size of the grant allocated in response to an ONU's *Report* message. However, the order in which ONUs are serviced is unspecified and is assumed to be round-robin. Note that the order may not necessarily be static, since IPACT may send grants in a different order in any cycle in an attempt to minimize the idle period of the upstream channel due to walk time to a farther ONU [71].

5.2.2 IPACT+SARF

We propose a new heuristic for use with the IPACT scheme. In fact, our heuristic is quite simple and independent of IPACT, and therefore could be used with any DBA scheme.

Figure 5-1 shows the SARF heuristic. SARF generates and responds to two events: $SEND\_GRANT$ and $REPORT_i$. A $REPORT_i$ event occurs when a *Report* message is received from ONU i. A $SEND\_GRANT$ event is generated by the SARF algorithm in step (4). Although our heuristic may be used with many different DBA schemes, we illustrate its use with IPACT in this work.

Initially, the OLT sends out, to all active ONUs, grants large enough to transmit a *Report* message (not shown in Figure 5-1). Next, whenever the OLT receives a *Report* message from ONU i, it first updates entry i in its table of current, known queue lengths and also marks ONU i as "pending". (A pending ONU is one which has transmitted a single new *Report* message, but has not yet been serviced.) Next, unlike plain IPACT, the SARF heuristic does not send out a grant to ONU i immediately in response to a report received from ONU i (except under a special condition described later). Instead, it defers grant transmission to the latest possible time. Suppose the *Report* from ONU i is received at time t. Let $S(t)$ denote the earliest time at which the upstream channel is known to become available as of time t. Then, SARF defers the grant transmission to a time t_g such that if a grant is transmitted at time t_g to

ONU i, it will cause the transmission from ONU i to arrive at the OLT exactly at time $S(t)$. Thus, SARF defers grant transmission to the latest time possible, without introducing any extra idle time (i.e., it maintains the "work-conserving" property of IPACT schedules). Clearly, a grant must be transmitted to an ONU sufficiently early so as to allow for the grant message transmission delay as well as the round-trip delay to the ONU. This is accomplished in step (4) of Figure 5-1. However, it may so happen that when a *Report* message is received at time t, $S(t) \leq d_i$ where d_i is the round-trip time to ONU i. In this case the grant message cannot be deferred since doing so would introduce unnecessary idle time (thus violating the "work conserving" property[1]). In this special case, the grant to ONU i is sent immediately.

When a $SEND\_GRANT$ event is triggered, the SARF heuristic is used to determine the sequence in which ONUs will be served. Of all the pending ONUs, the one with the smallest queue length is selected. Using a Gate message, the ONU is served a transmission grant. (Notice that the heuristic is independent of the policy used for deciding the grant size.) The status of the serviced ONU is changed from pending to "served". The ONU with the next smallest queue length is served next. This process continues until all pending ONUs have been served, exactly like the plain IPACT scheme. This completes one cycle of service and the same behavior is repeated for the next cycle. This is the basic description of the SARF heuristic when used in combination with the IPACT scheme.

5.2.3 Zero-length queues

Although the SARF heuristic chooses the ONU with the smallest queue length for service, it treats ONUs with zero queue lengths differently. An ONU with a zero queue length has no data to transmit. Under low loads, without special exception,

[1]We note that violating the work-conserving property may not necessarily be a bad idea [58].

such ONUs will always be served first. Serving zero-length queues still requires the allocation of a grant to accommodate the next *Report* message as well as the mandatory guard band overhead. (Guard bands are small intervals of time inserted between two consecutive grants to two different ONUs to prevent any possible overlap of transmissions due to small errors in time synchronization at those ONUs.) While such data-less grants do not contribute to lowering the packet delay at the source ONUs which have no packets to send, they do increase the delay faced by the succeeding ONUs. Hence, when choosing an ONU to serve next, ONUs with zero-length queues are treated as if they have a queue length that is equal to the average queue length taken over all ONUs. Moreover, we weigh this average by the number of times an ONU reports a zero length queue, consecutively. Thus, an ONU which has reported a zero length queue many times consecutively will likely be served at the end of the cycle.

5.2.4 SARF Rationale

The delay faced by any packet p in the EPON consists of three components: the reporting delay, the grant delay and the intra-grant delay. The reporting delay is the time between the arrival of a packet p at an ONU and the time at which it is counted and reported by the ONU to the OLT. The grant delay is the time between reception of a report by the OLT and the reception of the first bit of data associated with the report by the OLT (i.e., beginning of the actual grant). The intra-grant delay is the time packet p at ONU i waits after the beginning of a grant for ONU i for other preceding packets to be transmitted. Notice that in IPACT, the grant delay depends on the distance of the scheduling endpoint from the current time. As per the SARF heuristic, if the OLT chooses to serve the smallest request, it will increase the SEI by the smallest possible value. Thus, the SARF heuristic minimizes the grant

Table 5.1: Terminology used in the IPACT+SARF algorithm.

Term	Description
SEI	Scheduling Endpoint Indicator signifying the earliest time at which a new transmission can be scheduled on the upstream
$REPORT_i$	Event indicating receipt of a *Report* message from ONU i
$SEND\_GRANT$	Event signifying the latest opportunity to make a grant decision
MAX_RTT	Largest round-trip time among all connected ONUs in seconds
$GATE\_LENGTH$	Duration of a Gate message in seconds

delay faced by packets. In turn, the average cycle length may also be reduced thus leading to a reduction in the reporting delay. We observe that the main result, that for identical release times, the Shortest Processing Time rule provides the minimum completion time, is well-known in the area of scheduling theory [92]. We leverage this knowledge and successfully apply it to the DBA problem. However, we also note that this may *not* be the minimal achievable delay under the IEEE EPON architecture; we are currently investigating more sophisticated approaches to this problem.

5.3 Related Work

We note that to our knowledge, the work by Kamal et. al [63] was the first to realize the value of deferring a grant decision in order to benefit from more information. In their Prioritized MPCP scheme, low priority grants are deferred to the latest possible time (without introducing extra idle time) allowing higher priority grants to be

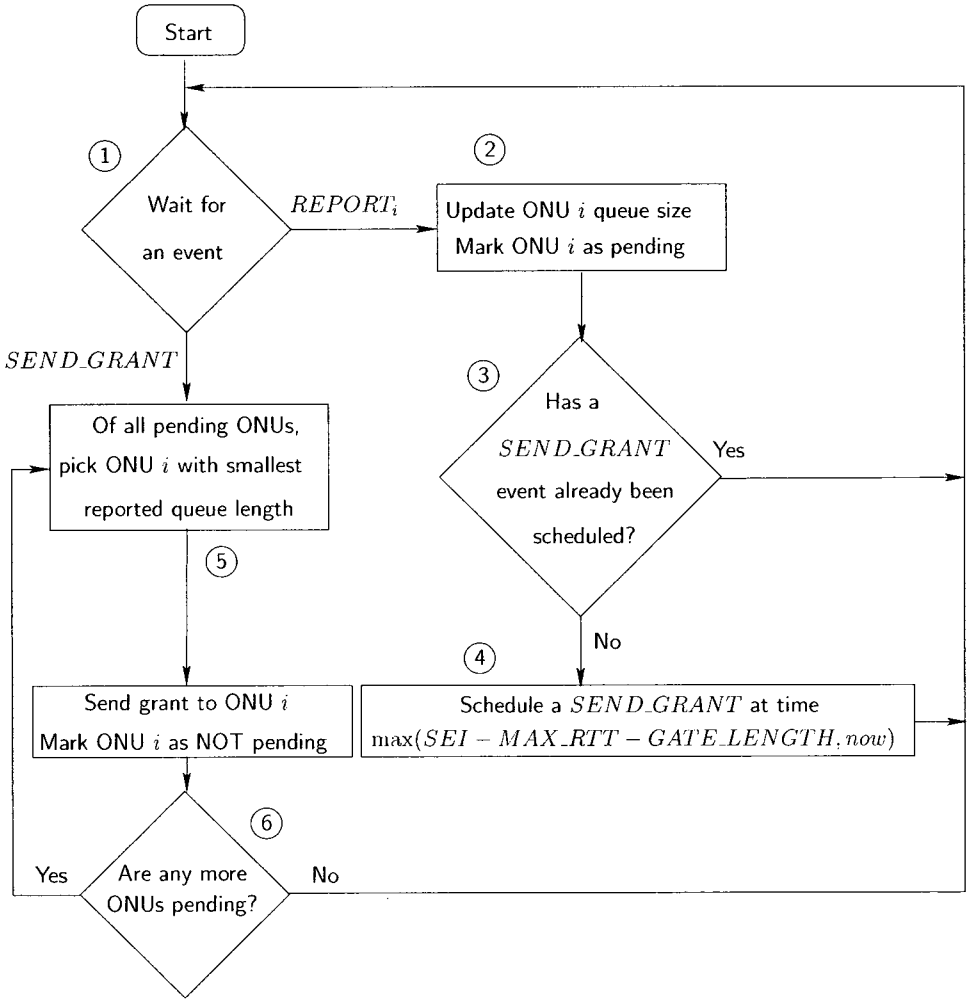


Figure 5-1: The SARF algorithm.

scheduled earlier. In this case, the order of granting is changed to enforce priority, not to minimize delay. However, Prioritized MPCP does suffer from other shortcomings such as priority inversion under certain load conditions as well as unfairness based on round-trip time to different ONUs.

Ma et al. [80] also propose a Dynamic Polling Order Arrangement (DPOA). However, for reasons not discussed in the paper, they order ONUs in descending order of the reported queue length. Their scheme (IPACT with DPOA) shows better performance than IPACT alone in the specific range of medium loads (0.5 to 0.8). The authors reason that the improvement in delay is due to better channel utilization. They also conclude that at high loads, since the channel is already heavily utilized, the polling order does not matter. However, their reasoning only holds for the conditions of their experiments (limited grants[2] and Poisson traffic), which are different from ours (gated unlimited grants and self-similar traffic).

We also note that the idea of differentiating between small and large requests (queue lengths) is also present in the work of Assi et al. [15]. However, the motivation for this in their scheme is slightly different (i.e., to maximize the overlap of walk time with transmissions in progress so as to minimize channel idling). We generalize this idea of differentiation based on request size in the sense that in the SARF heuristic, the size of the request (queue length) determines the exact position of an ONU in the polling sequence.

[2]Their paper [80] does not provide details about the grant policy used in their simulation experiments.

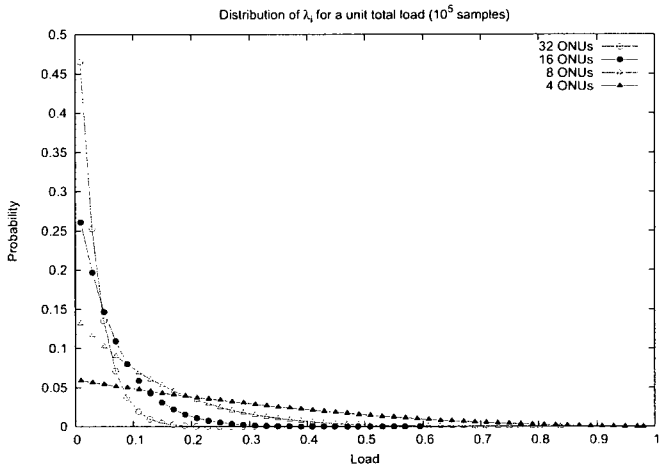


Figure 5-2: The marginal probability of an ONU load as measured from simulations.

5.4 Simulation Results

5.4.1 Simulation Details

We implemented the SARF heuristic with the IPACT scheme under gated-allocation policy, i.e., the OLT always allocates a grant exactly equal to the reported queue length with a fixed additional amount to accommodate the next *Report* message. In our simulation, we distributed the total load λ randomly across the $N \in \{4, 8, 16, 32\}$ ONUs. We achieved this using the following method: Given a total load $0 \leq \lambda \leq 1$, first pick uniformly at random, $N-1$ non-decreasing real numbers $0 \leq r_i \leq \lambda$, $i \in \{1, \cdots, N-1\}$. Then, assign the value $\lambda_i = r_i - r_{i-1}$ as the load for ONU i, with $r_0 = 0$ and $r_N = \lambda$. In this way, we assigned a random and hence nonuniform load to the N ONUs in each experiment. Generated in the manner described above, the random load λ_i on any ONU follows a Beta distribution $\beta(1, N)$ [37]. Figure 5-2 shows an example of the probability distribution of individual ONU loads when total load $\lambda = 1$. As illustrated, the chance of generating a load λ_i at ONU i is highly skewed in favor of lower loads. The traffic workloads for simulations were generated using a self-similar model with a measured Hurst parameter of approximately 0.8 [73]. Each

simulation was allowed to run for 10 seconds of simulation time. With self-similar traffic, simulations should be run for a much longer duration. However, self-similar trace generation and the simulations themselves are both computationally intensive tasks. Therefore, we report preliminary results with shorter simulations. Longer, more rigorous simulations are currently in progress. At least 40 simulation runs were conducted for each 0.05-length interval of load. (The exact number is difficult to fix due to the inherent error in generating the exact load with a self-similar traffic generator for a relatively short trace.) The guard band was set to 2 μs.

5.4.2 Results

Figure 5-3 shows the relative performance of IPACT with and without the proposed SARF heuristic. We plot the actual total load on the x-axis. On the y-axis, we plot the relative reduction in the average per-packet delay. This is calculated as

$$\Delta = \frac{\delta_{IPACT} - \delta_{SARF+IPACT}}{\delta_{IPACT}}, \tag{5.1}$$

where δ_{IPACT} is the delay of the plain IPACT scheme and $\delta_{IPACT+SARF}$ is the delay of the IPACT scheme combined with the SARF heuristic. Note that in each experiment, the traffic trace used as workload to evaluate the IPACT scheme was the same as the trace used to evaluate the IPACT+SARF scheme (i.e., IPACT was run with and without the SARF heuristic on the same trace). The "scatterplot" of points shows all actual measurements of the reduction in the delay, i.e., it shows 40 points per 0.05-long interval of load, each being a result of a 10-second long simulation. Each of these points is plotted at the actual load generated by the trace. For example, if the intended target load was l and the actual load generated by the trace generator was $l+\epsilon$, then, the relative reduction in delay measured (say r) is plotted as $(l+\epsilon, r)$ and not (l, r). Here, ϵ can be considered as the error in generating the required load. Since ϵ is unpredictable, there is no simple way of plotting an averaged curve that captures

the main trend. Therefore, we "binned" the delay reduction measurements as follows. We selected a bin size b ($b = 0.05$ in case of Figure 5-3) and created bins (intervals) B_k of the form $[b \cdot k, b \cdot (k+1)), k \geq 0$. Any measurement $(l + \epsilon, r)$ was dropped into the bin B_k where $k = \lfloor (l+\epsilon)/b \rfloor$. All measurements in each bin were averaged and the resulting average difference r_k was plotted as $(\hat{l}, r_k)$ where $\hat{l}$ is the average of all the loads in bin B_k. Clearly, our proposed SARF heuristic shows significant improvement over plain IPACT across most load values. For very low loads, it is likely that service sequence will not make a difference since the channel may be underutilized. Other cases where the SARF heuristic performs worse than plain IPACT may be explained by the observation that the SARF heuristic may result in many more, but smaller grants resulting in a somewhat larger overhead in the form of guard bands and *Report* messages. Figure 5-4 shows that the relative channel utilization of the IPACT+SARF scheme is very close to that of the original IPACT scheme.

5.5 Fairness

As discussed in Sec. 5.1.2, the issue of fairness in bandwidth allocation to ONUs in an EPON has received much attention in EPON DBA research [16][15][69]. In the proposed SARF algorithm, the choice of the next ONU to serve is guided solely by the reported queue size. Thus, an ONU with a smaller reported queue size may always be served before an ONU with a larger queue. This may be unfair according to some definitions of fairness [17]. Figure 5-5 illustrates this unfairness manifesting in the form of increased variance in the average of the average packet delay faced by the N ONUs under the SARF heuristic as compared to IPACT. To this criticism, we offer the following two responses. First, our approach is in stark contrast to other DBA approaches which seek to include fairness as one of the main requirements of the DBA algorithm. In our approach in this work, this is not our main concern. Instead, we

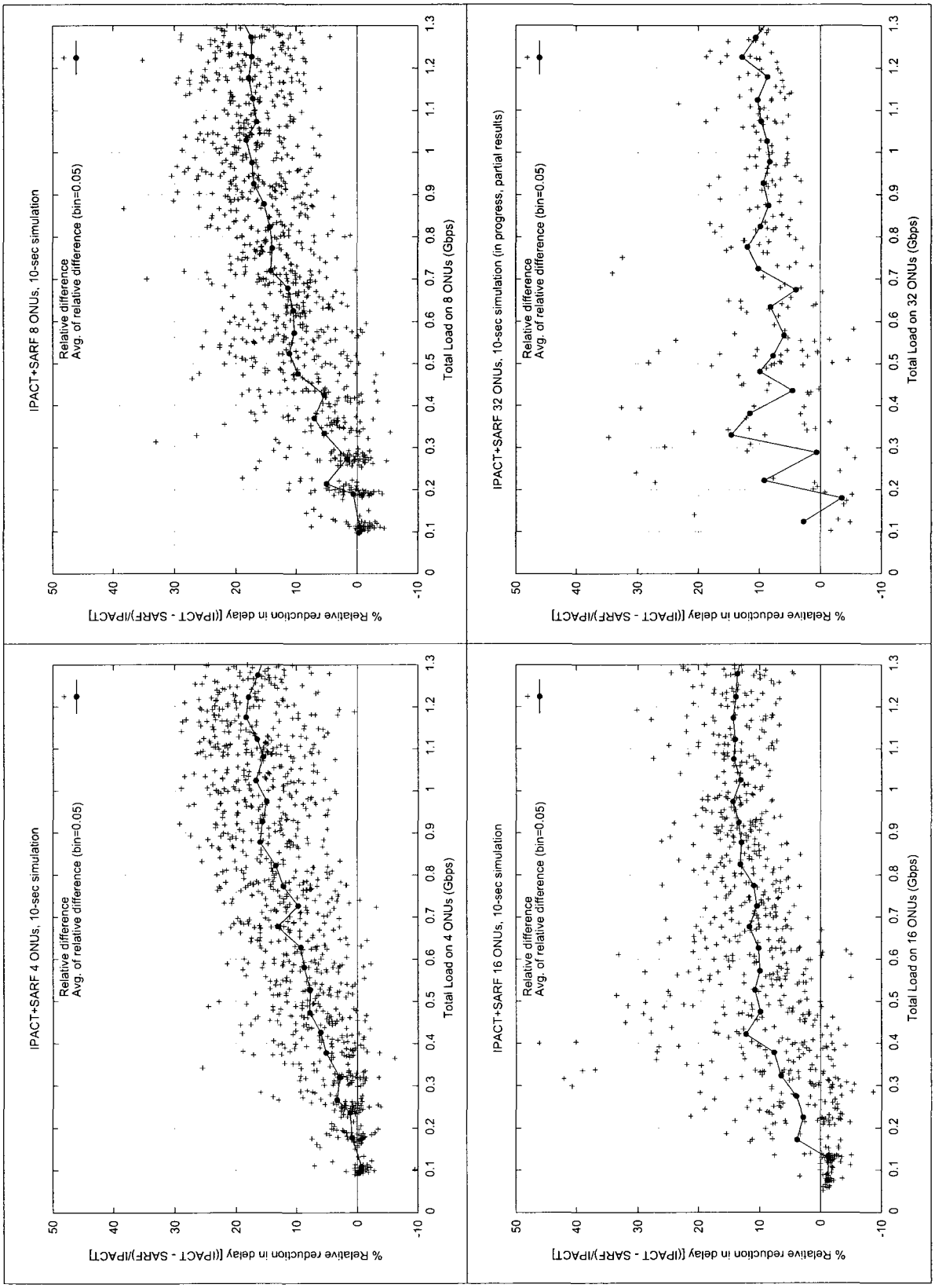


Figure 5-3: Comparison of mean packet delay of IPACT against that of IPACT combined with the proposed SARF heuristic.

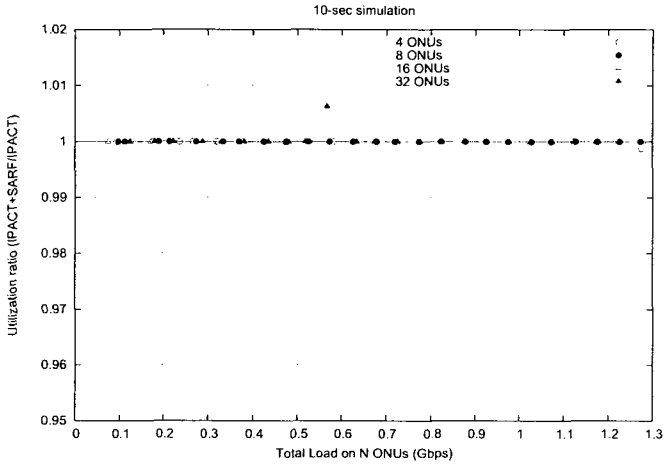


Figure 5-4: The utilization ratio of IPACT+SARF/IPACT.

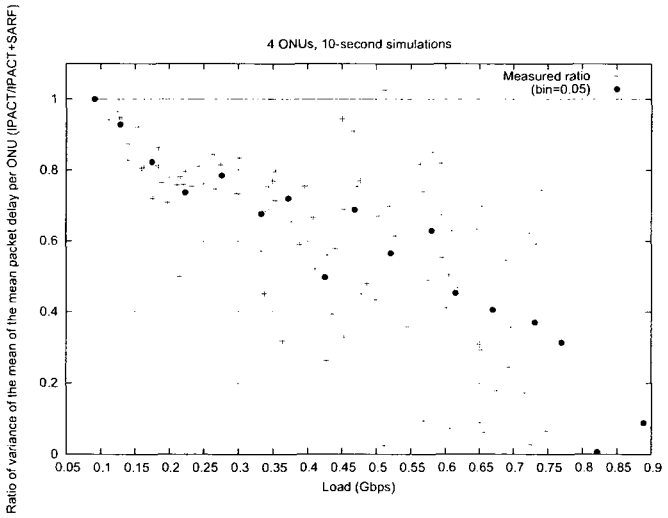


Figure 5-5: The ratio (IPACT/IPACT+SARF) of variance of the average (taken over all ONUs) of the average packet delay (taken over all packets per ONU) measured by a total of 50 10-second long simulations (preliminary results for 4 ONUs only).

focus on developing a DBA algorithm which, *by design*, attempts to minimize the overall average packet delay in the EPON. If that entails treating ONUs unfairly, then we allow the algorithm to make this intelligent decision. We believe that this is the main novelty of our approach to DBA algorithm design. Second, an element of fairness can be incorporated into our algorithm in a simple way. Instead of choosing the smallest report, one could easily choose the weighted smallest report and use the weights to provide fairness to ONUs.

5.6 Conclusion and Future Work

The main contribution of this work is the idea of exploiting the order in which grants are allocated to ONUs to minimize per packet delay. We proposed a new heuristic for the DBA problem which allocates grants using the Shortest Available Report First strategy. Our heuristic is independent of the actual DBA scheme and may be used in other DBA algorithms. We demonstrated its effectiveness using IPACT under the gated allocation policy. Our heuristic improves the delay performance of IPACT by about 10-20%. The SARF heuristic shows considerable promise and may be extended and improved. This work is currently in progress. An idea similar to SARF may be useful in discovering the optimal DBA algorithm possible under the IEEE EPON architecture. We are currently exploring these ideas.

Chapter 6

Empirical Evaluation of Upstream Throughput in a DOCSIS Access Network

We present empirical measurements of the upstream throughput of a DOCSIS[1] 1.1 link. In contrast to all previous simulation-based studies, our measurements have been obtained from actual cable-modems (CMs) and head-ends, both from two different vendors each. We have constructed an exhaustive database of measurements of a large subset of the space of parameters affecting upstream throughput. Using a well-known non-parametric hypothesis test, we query this database for obtaining statistically robust answers to key questions about the effect of parameter changes on the throughput. Our results indicate that for a single CM scenario, packet concatenation is most effective whereas piggybacking is effective and better than concatenation only in some cases. Using both enhancers decreases throughput for a single CM scenario. Our results are robust across head-end implementations and are of immediate interest to network and protocol architects as well as device developers.

[1]DOCSIS is a registered trademark of CableLabs.

6.1 Motivation

The DOCSIS 1.0 and 1.1 standard [1] was a result of research and development into the performance implications of various QoS mechanisms [101, 33, 53, 29, 40, 77]. Since these were preliminary, prototyping studies, they were conducted using analytical and/or simulation models. There has been no extensive study of the DOCSIS protocol using actual implementations, to our knowledge. However, studies based on actual implementations provide a wealth of complementary information of interest to designers and operators alike. Measurements from an implementation offer a tractable way of capturing and evaluating a system in its entirety. For network service providers and operators, empirical data obtained from actual implementations is an indispensable input for the design and upgrade process. Providers and operators require some form of field performance report from device vendors, before a device can be deployed in a live traffic environment. Needless to say, an empirical model must be tested for robustness using standard statistical techniques. Without such testing for confidence, no useful and reliable conclusions can be drawn from the data.

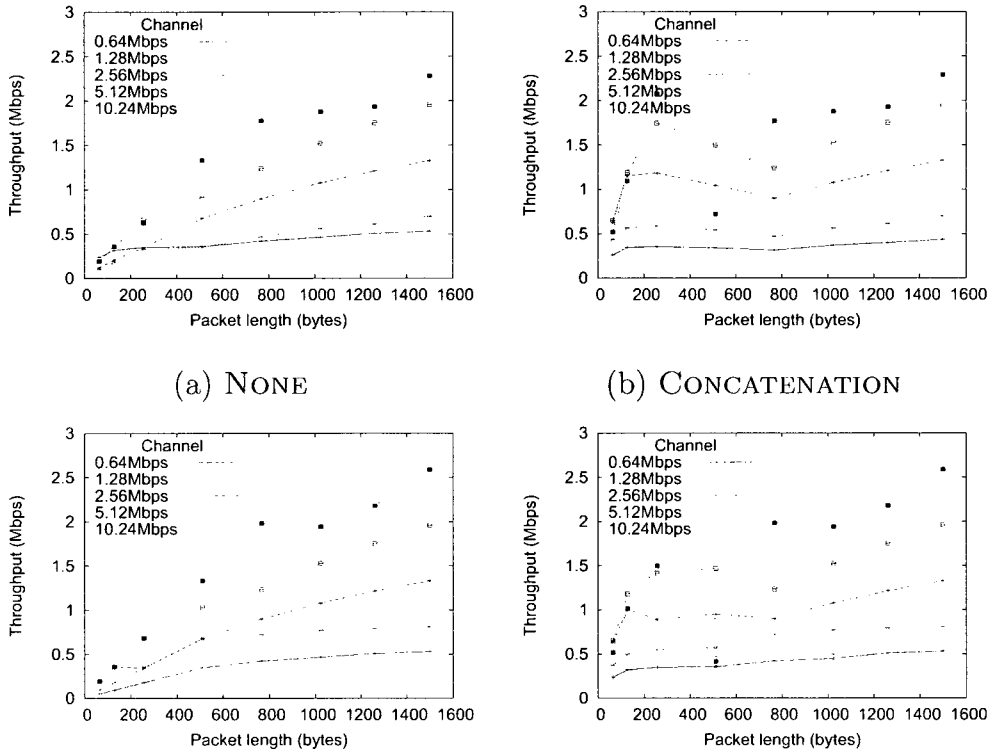
In this work, we report results from our extensive performance study of the upstream portion of a DOCSIS 1.1 link based on real devices. Using our measurements as a database, we can answer key performance questions that are of immense value to the network designer. We also hope that the results obtained provide some insight into the strengths and shortcomings of the various performance enhancing features provided by the protocol.

6.2 An Overview of the DOCSIS 1.1 MAC Layer

A DOCSIS network uses the existing cable television infrastructure to deliver data services to subscribers. The network, owing to the structure of the pre-existing cable television plant, forms a tree with the root connected to a Cable Modem Termina-

tion System (CMTS). Subscribers connect to the network through a cable modem (CM) connected as a leaf of the tree. The link from the CMTS to CM, termed as downstream, is point-to-multipoint broadcast, whereas the upstream from the CM to CMTS is a multipoint-to-point time-division multiplexed link arbitrated by the CMTS. The upstream channel can transmit at any pre-configured rate from the set {0.64, 1.28, 2.56, 5.12, 10.24} Mbps. The downstream is also configurable, but is not the focus of this work. The CMTS describes the allocation of the upstream bandwidth for a future interval of time using a map message regularly broadcast downstream. In each mapped interval, the CMTS reserves portions of the upstream bandwidth for new CM registration and bandwidth requests from existing CMs. A CM wishing to transmit data on the upstream first requests bandwidth by transmitting a message during the bandwidth request interval, then waits to receive a bandwidth grant in a map message, and finally transmits data during its map-designated time slot.

The bandwidth request interval is contention-based and is therefore prone to collisions from overlapping request messages from many different CMs. Therefore, as an alternative to the contention-based request interval, a CM can also request bandwidth during a data transmission opportunity, i.e., it can use part of the transmission interval acquired for data transmission through contention-based requesting to make further requests for bandwidth. This is known as *piggybacking* and is one of the enhancements that can be provided to or prohibited for a CM through a configuration file. Piggybacked bandwidth requests use a small part of the data bandwidth to make further requests and thus avoid delays in acquiring bandwidth due to lost request messages. In addition to piggybacking, *packet concatenation* is another available enhancement. In order to minimize overhead and reduce latency, a CM can send a longer burst of concatenated packets in a single transmission opportunity instead of stepping through the request-grant-transmit cycle repeatedly for a sequence of small



(a) NONE (b) CONCATENATION

(c) PIGGYBACKING (d) BOTH

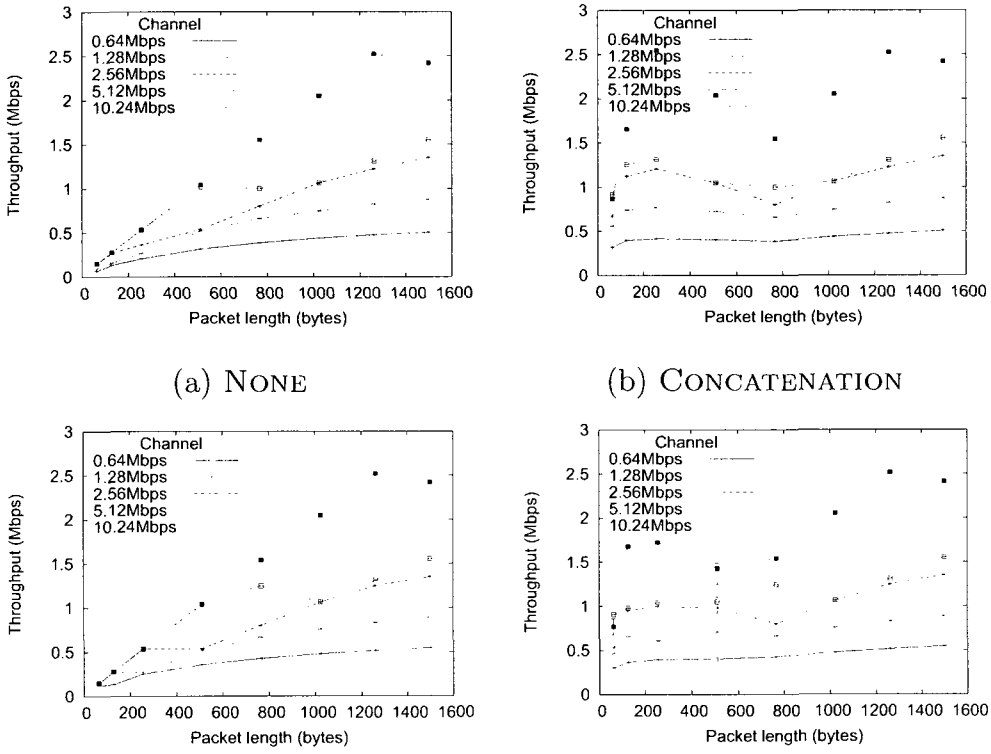
Figure 6-1: Throughput of CM_A on $CMTS_C$ with No Enhancers, Concatenation, Piggybacking and Both enabled respectively (99% confidence).

packets.

In this work we focus on characterizing the performance of the two enhancers as well as the effect of the channel rate and packet length on the throughput. We describe our experiments in the next section.

6.3 Experimental Design

The experiments presented in this work were conducted on the testbed shown in Figure 6-3. The test network consisted of a CMTS connected by a coaxial cable plant to one or more CMs. Upstream traffic was generated using a traffic generator, injected into the cable plant through the CM, and captured beyond the CMTS by a



(a) NONE (b) CONCATENATION

(c) PIGGYBACKING (d) BOTH

Figure 6-2: Throughput of CM_A on $CMTS_D$ with No Enhancers, Concatenation, Piggybacking and Both enabled respectively (99% confidence).

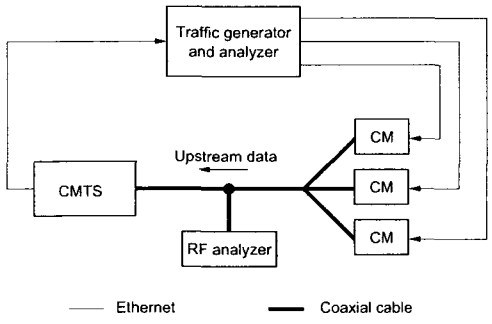


Figure 6-3: Experimental setup.

traffic analyzer. The distance between the CMTS and the CM was negligible (a few feet).

Our experiments were conducted in two phases dictated primarily by the availability of equipment for the necessary duration. Access was available to several units of two different CM implementations (CM_A and CM_B) and one unit each of two different CMTS implementations ($CMTS_C$ and $CMTS_D$) individually, at different times. In the first phase, we conducted pilot experiments involving CM_A and CM_B against a single $CMTS_C$. In this phase, the experiments spanned a broad range of parameters and were intended to provide the big picture. Effect of packet length, channel width, offered load, modulation format, number of CMs and enhancers on latency and throughput was studied. These results are reported in [19, 52, 2]. Based on some of the preliminary findings, we devised new experiments to study interactions between parameters and the throughput at a finer level.

In the second phase, we characterized the effect of every change in the system parameters on throughput. The second phase was designed to be more exhaustive and is reported in this chapter. Data was collected using CM_A against $CMTS_C$ and $CMTS_D$. Data was injected into the CM at a constant input rate of 8 Mbps since this was within the saturation region as characterized in phase 1 experiments.

In each experiment, the system was configured to provide the CM with a different combination of channel rate, data packet length and performance enhancer. Different packet lengths from the set {64, 128, 256, 512, 768, 1024, 1262, 1500} (bytes) were used. Finally, all four different performance enhancer combinations (packet concatenation, piggybacking, neither and both) were applied. We also considered the CMTS ($CMTS_C$ vs. $CMTS_D$) as the fourth system parameter since we were interested in obtaining performance measures that were robust across CMTS implementations. The space of all the possible combinations of parameters was explored. Across two CMTS and with 5 channel rates, 8 packet lengths and 4 performance enhancers, a total of 2400 (1120 per CMTS and 160 between CMTS) different transitions from one set of parameters to another leaving the values of all but one parameter unchanged can be performed. For each case, 25 independent observations were recorded to ensure high confidence in the conclusions derived. Thus, for each experiment, two sets of 25 throughput values were obtained—with and without the change in value of one parameter.

In order to draw robust conclusions from our data, we used the Wilcoxon Signed Rank Sum (WSRS) test [98] for testing the effect of each of the 2400 transitions from one parameter set to another. The WSRS test is a commonly used non-parametric hypothesis test when the distribution of the data are unknown and is able to deal with paired data. Given a set of ordered pairs $\{(x_i, y_i)|1 \leq i \leq N\}$, where x_i is the response prior to or without the modification whereas y_i is the response with the modification (or vice versa), the WSRS test computes the probability that the median of the distribution of $x_i - y_i$ is zero (i.e. the distribution of sorted differences is symmetric about zero). This is the *null hypothesis* of the WSRS test and represents the scenario where the modification or solution is ineffective. If this probability is lower than the preset *significance* level, then the test concludes that the null hypothesis of inefficacy

of the solution can be safely rejected since such a rejection is amply supported by the available data. To apply the WSRS test to each experiment, we use the null hypothesis that the change in the value of one of the system parameters produced no effect on the throughput. We fix an acceptable level of significance to 95% ($\alpha = 0.05$). Since we compare many different modifications (the 2400 different parameter transitions) we perform 2400 pairwise tests. In this scenario, although each hypothesis taken separately is significant at level α, all the hypotheses taken together are not significant at level α. In order to achieve a significance level of α for all the tests taken together, we apply the Bonferroni correction [98] and require each of the 2400 tests to be significant at a level $\frac{\alpha}{2400} = 2.08333 \times 10^{-5}$ (i.e., 99.99% significance individually). The parameter-set pairs with their significance levels were recorded into a database.

6.4 Discussion of Results

Figures 6-1 and 6-2 show the throughput of a single CM_A on $CMTS_C$ and on $CMTS_D$ respectively. The results are all presented with 99% confidence intervals[2]. Comparing the two figures, we see that the throughput on both the CMTS shows a similar trend in each case. The drop in throughput for $CMTS_C$ for a packet length of 512 bytes cannot be explained. Given that the data point passes the confidence test, we speculate that it may be related to an implementation bug. Although not as large, comparing plot (C) in each figure shows the improvement in throughput for larger packet sizes when piggybacking is enabled. Finally, plot (D) in each figure shows the throughput with both enhancers enabled. Variations in throughput within each figure may be of interest to network designers and operators whereas variations across the same plot in the two figures may be of interest to the CMTS vendors. It is important

[2]The 99% confidence intervals are plotted in the graphs but may be too small to be visible at most of the points.

to keep in mind that since the data points presented are bound by tight confidence intervals, very small differences in performance are still statistically significant.

We now turn our attention to the relative improvement in throughput offered by various combinations of enhancers. Table 6.2 (C) shows throughput as a function of packet length for various enhancers. Clearly, for smaller packet sizes, concatenation proves to be extremely effective. Moreover, it should be noted that concatenation together with piggybacking is less effective than concatenation alone. This can be attributed to the fact that for a single CM, piggybacking provides no benefit for smaller packet sizes, but does consume a small portion of the bandwidth provided. However, piggybacking provides a slight improvement in throughput for larger packet sizes where concatenation does not seem to have a pronounced effect. Table 6.2 (D) shows the change in channel utilization with enhancers for different channel rates. Again, concatenation is most effective, more than piggybacking alone or in combination.

Based on these initial observations, we can query the measurement database for some more specific questions. For example, it would be of interest to network designers to know, for an actual system to be deployed, when and by how much piggybacking improves throughput. Table 6.2 (A) answers this question with 95% significance for a single CM scenario. To answer the question, we pick all those transitions of parameter values from our measurement database in which the initial parameters have no enhancers enabled whereas the final parameters have only piggybacking enabled. Moreover, we only pick those points satisfying this condition which show the same trend—either an increase or decrease in throughput—across both CMTS at the required significance level of 95%. For the current question, we find statistically significant improvements only for the points shown in Table 6.2 (A). For the remaining cases, either piggybacking is not useful or there is not enough evidence to conclude so with 95% significance. Table 6.2 (B) answers the related question if, when and

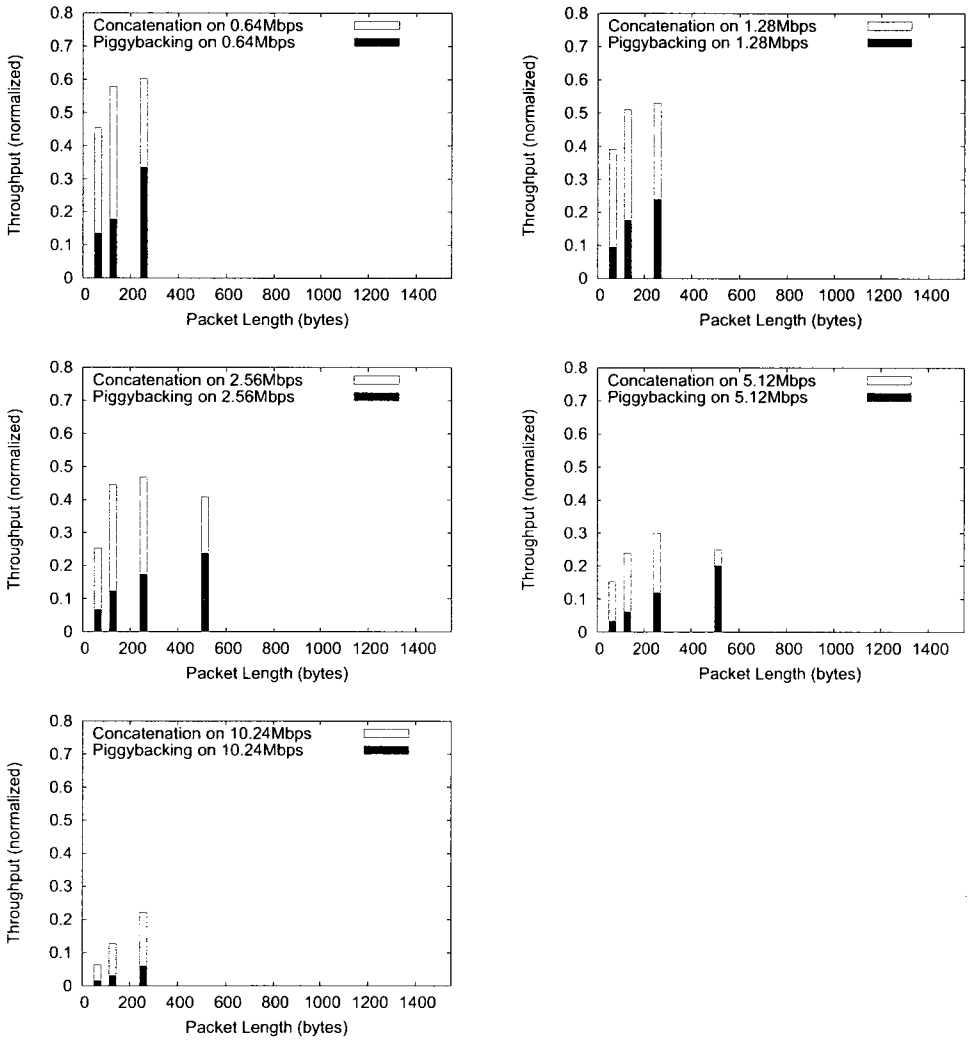


Figure 6-4: When and by how much is Concatenation better than Piggybacking? (95% significance, points failing this criterion omitted.)

Table 6.1: (a) When and by how much is Concatenation better than Piggybacking? (b) When and by how much is Concatenation useful? (c) When and by how much is Piggybacking and Concatenation worse than just Concatenation?

(95% significance, points failing this criterion omitted.)

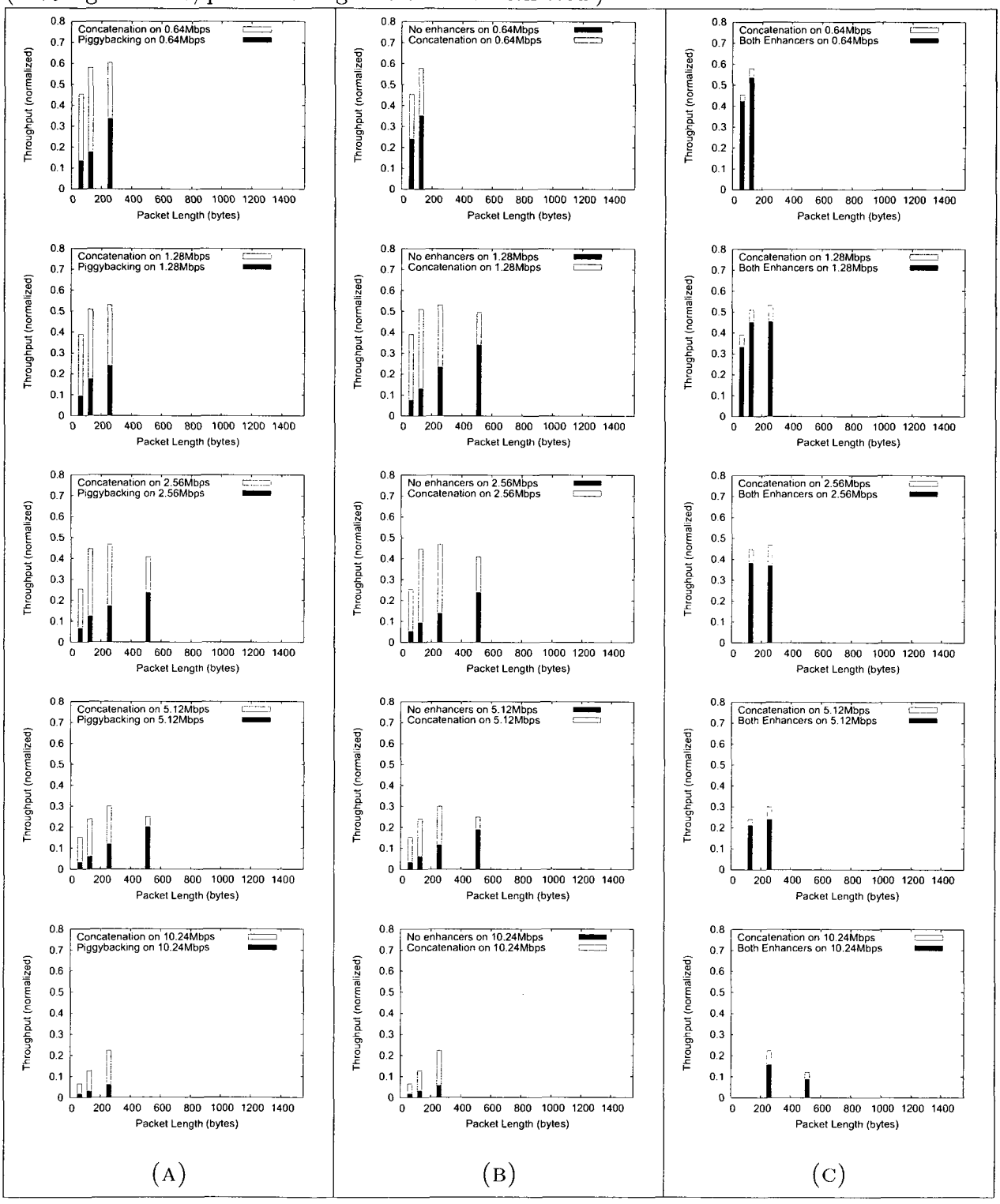
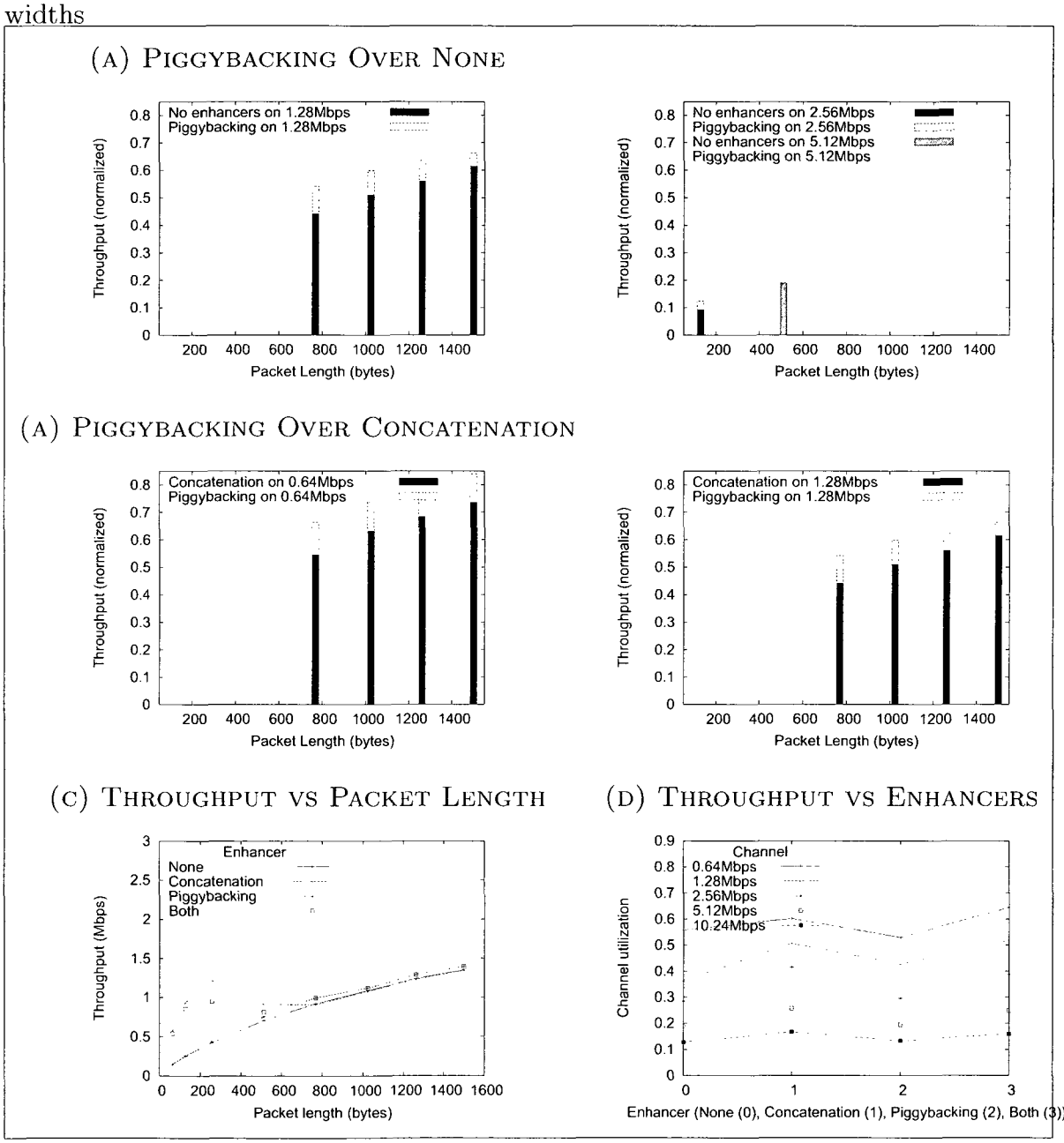


Table 6.2: (a) When and by how much is Piggybacking useful? (b) When and by how much is Piggybacking better than Concatenation?
(95% significance, points failing this criterion omitted.) (c) Throughput vs. Packet Length for various Enhancers (d) Throughput vs. Enhancers for various Channel widths



how much better piggybacking performs compared to concatenation. We find that for larger packets and smaller channels piggybacking is capable of improving throughput even in the single CM scenario by as much as 10-20%[3]. Table 6.1 (A) on the other hand answers the complementary question: when and by how much is concatenation better than piggybacking? As alluded to by the results in Table 6.2 (C), concatenation proves much better than piggybacking for smaller packet sizes. Table 6.1 (B) illustrates the cases where throughput with concatenation is higher than that with no enhancers. Again smaller packet sizes are more amenable to concatenation. Finally, Table 6.1 (C) asks the question if, when and by how much throughput is lowered by using both enhancers instead of concatenation by itself. We find that throughput can be lowered by as much as 10-15% for smaller packet sizes if both enhancers are used simultaneously in a single CM scenario. The questions asked above are only a small sample—many other such answers can be mined from the database.

6.5 Conclusion

We have conducted an extensive evaluation of the throughput of an actual DOCSIS upstream link. We report the results of our measurements and based on the data collected answer some questions of interest to network designers and operators as well as device developers. We conclude that concatenation is most effective for small packet lengths but is outdone by piggybacking for larger packet sizes and lower channel rates. In general, even in the single CM scenario, piggybacking is useful for some packet lengths across medium sized channels. We also find that at least in the single CM scenario, concatenation alone works much better than in combination with piggybacking. Our conclusions are robust across two different CMTS implementations

[3]All percentage changes reported reflect the trend. Real values vary widely from a few to a few hundred percent.

and are acceptable with 95% significance. The WSRS test is well-suited for testing significance of differences between alternative schemes.

6.6 Future Work

The second set of our experiments is still incomplete. For example, piggybacking must be studied with multiple CMs. In general, the performance of the system in multi-CM scenarios remains largely unexplored. In addition to these simple performance enhancers, DOCSIS also provides a set of guaranteed flows such as unsolicited grants (guaranteed bandwidth), polled real time and non-real time grants (for voice and video traffic). Two other schemes, payload header suppression and fragmentation also remain open for study.

Part II

Mobile sensor networks

Chapter 7

Introduction to Mobile Sensor Networks

Falling costs have enabled the construction of low cost sensors equipped with the ability to sense, compute, and communicate. It is now possible to deploy such sensors in large numbers and obtain data at a finer spatial and temporal resolution than before. While sensors can independently acquire data from their immediate environment, delivering acquired data to a central repository for processing and decision-making requires that sensors communicate and transport data cooperatively. Research in sensor networks, which has seen a lot of activity in recent years, proposes and studies protocols and algorithms for the aggregation and routing of data in such collections of sensors in an energy efficient manner.

The sensors in such networks are assumed to be placed in an environment for a relatively long period of time, the length of which is dictated by the longevity of the energy source of each sensor. During their period of deployment, except for accidental environmental disturbances, the absolute geographical location of each sensor and its location relative to other sensors remains fixed. Many phenomena to be measured by the sensors, however, are not geographically static. For such applications, mobile sensor networks are a more appropriate solution. In such networks, each sensor is mounted on a host that is capable of locomotion.

Such sensor mobility comes in two varieties: programmable and natural. By

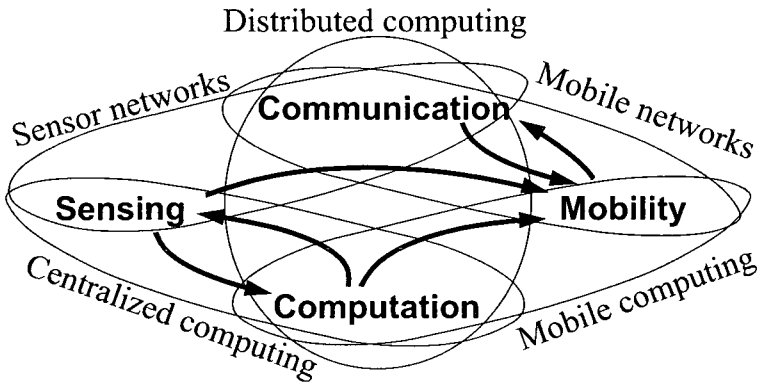


Figure 7-1: A computational mobile sensor compared to other types of networks and how mobility, computation, communication, and sensing interact in such a network.

programmable mobility, we mean that the sensor can control and dictate the mobility of its host. Sensors mounted on autonomous vehicles or robots are examples of programmably mobile sensors. In contrast, by natural mobility we mean hosts which are not under the control of the sensor. For example, sensors attached to fauna, or dispersed in moving water or wind, or piggybacked onto vehicles on a fixed course, exhibit natural mobility. Mobility that is a hybrid of these two types is also possible.

Past work has mostly been in the area of static sensor networks and—except for the work of Giridhar [49, 50] and Kowshik [67]—has focused on the efficient transport (aggregation) of acquired data [118, 119, 110]. A central assumption of our work is that a computational mobile sensor network is deployed for a specific purpose which we call its *mission*. A mission is a computational task whose input comes from the environment. The output of the computation could serve as input to another mission or could be utilized by the network to act upon its environment. In this setting, mobility, computation, sensing, and communication are related to each other as shown in Fig. 7-1. The mission dictates the sensing and computation to be undertaken by the sensors. If the sensor network is programmably mobile, then this in turn dictates the pattern of mobility followed by the sensors. Mobility,

however, also impacts the ability of sensors to communicate with each other. The reliability of wireless communication is inversely related to the distance between the sensors; depending on the transmission medium, the product of the range of reliable communication and its bandwidth can be severely limited [93, 36, 28]. The mobility induced by the sensing requirements of the mission may force sensors to drift out of each other's communication range whereas the mission computation may require a sensor to accumulate data from distant peers through multi-hop communication while continuing to contribute data through its own sensing. Thus, mobility must serve the needs of two possibly conflicting tasks: sensing and communication.

This introduces a new constraint on the communication model available to each sensor. Unlike the static network case, a mobile sensor cannot rely on the ability to communicate with a specific peer—or any peer at all—on demand. Rather, the sensor must await and exploit short-lived opportunities to communicate with peers in its vicinity. If the sensors are programmably mobile, then their motion can be choreographed to support the overarching computational mission. In Chapter 8, we propose a design that achieves this through a precomputed schedule of meetings among sensors, which we call a *tour network*. Each sensor is assigned a tour, i.e., a schedule, that includes time for the sensor to complete its independent sensing task and arrive at a meeting point to share its data with some of its peers. The design allows for mechanisms to detect and react to contingencies such as sensors that fail to appear at a scheduled meeting point. While there has been work on designing trajectories for mobile routers [119], ours is the first to propose an architecture for tours and their transformation in response to environmental challenges in a distributed way.

The above discussion applies to the case where the sensor nodes are programmably mobile. Angluin et al. [12] have proposed a model of distributed computation for naturally mobile sensor networks. Their model, called a population protocol, is based on

the assumption that naturally mobile sensors are nonintrusive (small) and inexpensive and therefore severely limited in storage and computational power. They show that if all sensor nodes are identical and anonymous, allowed only a finite amount of memory, and mobile in a way that allows every pair of sensors an opportunity to interact with each other, then the class of solvable missions is restricted to the evaluation of semilinear predicates. Concurrently but independently, Chandy et al. began the study of functions that are computable in dynamic distributed systems of computing nodes. In this setting, arbitrary groups of nodes are formed temporarily and repeatedly, and the participants opportunistically make progress on the computation. We call such functions *self-similar functions* (after their similar nomenclature for algorithms that can execute correctly in such conditions). These two models of computation in naturally mobile systems of nodes are similar in their motivation and assumptions. In Chapter 9, we study the relationship between these two models.

The computational power of population protocols is known when every pair of sensors is allowed to interact with each other. When the allowable interactions are constrained, not everything about their computational power is known. One approach to studying population protocols is to view their state spaces as directed graphs and study what properties of these graphs reveal about the computational power of population protocols. A population protocol comprising n nodes and m states can be represented as a directed graph on m^n vertices. For all n, one can think of this graph as being generated by the population protocol, itself a digraph on m^2 vertices. It is plausible that one can generate the digraph on n nodes by defining an operation on the digraph on m^2 nodes. This leads us to the study of product operations on graphs in general. In Chapter 10, we study such operations in general and show that some invariants of product graphs can be uncomputable. The question of representing state spaces of population protocols as digraph powers and their study remains open.

Finally, in Chapter 11, we experimentally study the throughput-delay tradeoff in *sparse* mobile sensor networks. Throughput and delay are two important measures of performance of a network. Gupta et al. [57] showed that in static networks, the throughput of the network goes to zero as a function of the node density of the network. This is primarily a result of interference between nodes that are all within communication range of each other. As density increases, while the number of nodes available for forwarding traffic increases, the total throughput (also known as the capacity of the network) does not increase at the same rate, and hence the total throughput goes to zero. This important result led to a line of theoretical work investigating throughput as a function of node density. Grossglauser et al. [55] extended this work to mobile networks showing that the result of Gupta does not hold when nodes are allowed to be mobile: mobile nodes can store packets that they are responsible for forwarding and only transmit them when they happen to find themselves close to the recepient of the packet. In this way, throughput that is constant as a function of node density is achieveable. The drawback of this scheme, however, is that the mobile nodes must delay transmission of packets until they are within communication range of the recepient. The time to this event depends on the mobility pattern of the nodes and can lead to unbounded delay. This led to new theoretical work investigating the tradeoff between throughput and delay as a function of node density under different mobility models. All of this work, however, assumes that at any given time a mobile node is within communication range of at least some other node. For sparse mobile networks such as the AUV network of our example, this is not true. To the best of our knowledge, the throughput-delay tradeoff for sparse networks has not been studied. This is our goal in Chapter 11. We investigate different packet relaying strategies in sparse mobile networks and their impact on throughput and delay.

Chapter 8

Choreographing communication in mobile sensor networks

We consider a mobile sensor network in which a collection of programmably mobile sensor nodes are charged with a mission. The mission involves sensing data from the environment in which the nodes are deployed and performing a joint computation on the data. We assume that the smallest dimension of the geographical sensing area assigned to each sensor node is large compared to the node's range of communication. Thus, our study is focused on mobile sensor networks that are *sparse*, where sensors are programmably mobile, and are deployed to execute a mission.

As discussed in the introduction, sensing and communication often require that the sensor node move in conflicting ways. Moreover, the transmission medium and the sparseness of the distribution of the sensors may make the network of sensors a completely disconnected collection of sensor nodes. Yet, there exist practical applications for such networks: underwater exploration, surveillance, and search using autonomous underwater vehicles (AUVs) is one such set of examples where the AUVs operate as isolated nodes for most of the duration of their mission. This is primarily because of the vastness of the operating region of such AUVs (usually of the order of hundreds of square miles) and the poverty of the bandwidth-distance product of the acoustic transmission medium, the best available communication medium under water. Thus, submerged AUVs must either repeatedly surface to communicate over

long distances using radio waves or must move within a few hundreds of meters of each other to be able to converse at a sufficiently high data rate. In a typical sensing or clandestine surveillance mission, often neither option is feasible.

We assume that the network of mobile sensors is deployed on a mission involving periodic sampling and joint computation. We postpone discussion of the actual computation during the mission to future work. Here, we only propose and discuss an abstraction that aids thinking about the networking problem as one of facilitating opportunities for the disconnected collection of sensors to interact with each other rather than one of sustaining the illusion of a connected network. It is our thesis that a more robust and efficient design for mobile sensor networks is possible via the former abstraction rather than the latter. Our new design abstraction for such collections of disconnected mobile sensors is called a *tour network*. In a tour network, each mobile sensor in the collection is assigned a *tour* comprising sensing and meeting, and meetings connect tours into a tour network.

8.1 Tours and tour networks

A sensor's tour represents its share of the sensing workload, its motion trajectory, and its opportunities for meetings with its peers, for one cycle of the periodic mission. A tour is defined by an *area*, a collection of *meeting points*, and a *schedule*. The tour area represents the geographical area assigned to the sensor: it is singly responsible for acquiring data within that area. A meeting point is a configuration of the system in which interaction between sensors participating in the meeting point becomes possible. In its simplest form, a meeting point is a geographical location in whose vicinity participating sensors gather to communicate with each other. The tour schedule defines the times at which the meetings of the sensor nodes occur at the given meeting points. Each sensor node is assigned a tour and it is said to implement

the tour if it follows a trajectory of motion that allows it to acquire data from its tour area and arrive at all its meeting points on time, repeatedly. We call one round of all the meetings of a tour a *cycle* of the tour. The number of meetings in a tour is called its *length.* A *mission cycle* is the longest cycle among all tours in the network implementing the mission. A tour may impose quality requirements on trajectories that implement it. For example, a tour may require that data be acquired from the tour area at a minimum rate. We postpone discussion of such quality metrics to future work. Finally, tours assigned to sensors that participate in a joint meeting point are said to be related to each other. From this relation over the set of tours, we obtain a network of tours.

Formally, a tour network can be represented as an undirected hyper-multi-graph: a loopless undirected graph on the set of tours in which edges may involve more than two vertices and the same set of vertices may participate in more than one edge. An edge in a tour network represents the fact that the participating tours (i.e., the sensors implementing the tours and thus, their corresponding regions) will be able to exchange data and perform joint computations at a given location periodically. The vertices and the edges of a tour network may be labeled with geometric data. For example, each vertex may be labeled with the tour area and each edge may be labeled with the spatial coordinates of the meeting point.

Figure 8-1 shows an example of a tour network. The network comprises 16 mobile sensor nodes named $(0,0)$ through $(3,3)$, and hence 16 tours or vertices. The tour area of sensor node (i,j) (and hence vertex (i,j)) is the square (i,j) according to the coordinates shown in the figure. Filled circles represent meeting points. Thus, the tour of sensor node $(1,1)$ contains four meeting points. The location of these meeting points is on the periphery of its tour area. Each of its meeting points is shared by exactly one other sensor node. The arrows between meeting points denote

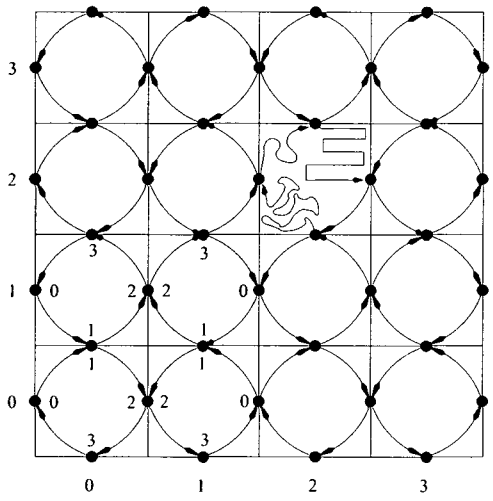


Figure 8-1: Example of a tour network comprising 16 mobile sensor nodes.

their temporal order in each cycle of the node's mission. The numeric label for each meeting point is the time of that meeting during its tour. We call this tour the *circle* tour. Suppose that each tour in Figure 8-1 is 4 hours long. Each sensor node must start its tour at the meeting point labeled zero. Thus, sensor node $(0, 0)$ must start its tour at the meeting point located on the west edge of its tour area. In the figure, this meeting point is not shared with any other sensor node. This may represent, for example, that the AUV sensor node must surface and communicate with the mission base at this meeting point, or that the results of the mission are fed into (composed with) another mission through this meeting point. After completing its interaction at the initial meeting point, the sensor node is free to engage in its sensing workload. Its trajectory of motion is unconstrained until the time of its next meeting, at which it must arrive at the next meeting point. (An example of the unconstrained trajectory followed by a sensor node between its meeting points is shown for sensor node $(2, 2)$ in the figure.) The next meeting of sensor node $(0, 0)$ is at time 1 (i.e., at the end of the first hour of its tour) and is located on the northern edge of its tour area. Unlike its previous meeting point, this meeting is shared by its peer node $(0, 1)$. The

sensor nodes must arrive at the meeting point at the same time. We assume that the nodes execute a simple beacon-based protocol to detect each other's presence at the meeting point in a short time window around the designated meeting time. The nodes then exchange data or execute joint computation using a simple TDMA protocol. Finally, nodes must signal the end of the meeting at which time each node continues on its sensing trajectory until its next meeting time. We acknowledge but postpone to future work a discussion of the scenario where sensor nodes fail during any of the various meeting protocols.

Figure 8-2 shows the time evolution graph of the example tour network shown in Figure 8-1. The m-step time evolution graph of a tour network $\mathcal{N} = (V(\mathcal{N}), E(\mathcal{N}))$ is a directed graph with the vertex set $V(\mathcal{N}) \times \{0, \ldots, m-1\}$ with an arc from (a, i) to (b, j) if and only if:

1. $a = b$ and $j = i + 1$, or

2. a co-participates in a meeting with b in $\mathcal{N}$ at time t and $t = i = j$.

A time evolution graph is a useful visualization of the tour network because it clearly shows the flow of information in a tour network. The first condition represents the fact that a sensor node carries forward in time all the information it has acquired at step i. The second condition represents the possible exchange of information during a meeting in which two sensors participate. An m-step time evolution graph shows the dissemination of information in m steps. The existence of a path from (a, i) to (b, j) in an m-step time evolution graph indicates that information available to sensor node a at time i reaches sensor node b by time j. Let (a, i) be a vertex in the m-step time evolution graph of $\mathcal{N}$. The set of all paths starting at (a, i) is called the *cone* of (a, i) and the set of all paths to (a, i) is called the *funnel* of (a, i). All vertices of the form (a, i), for a fixed i, form *slice* i and for a fixed a, form *layer* a. The cone (funnel) of a layer a (slice i) is the union of the cones (funnels) of all vertices in layer a (slice

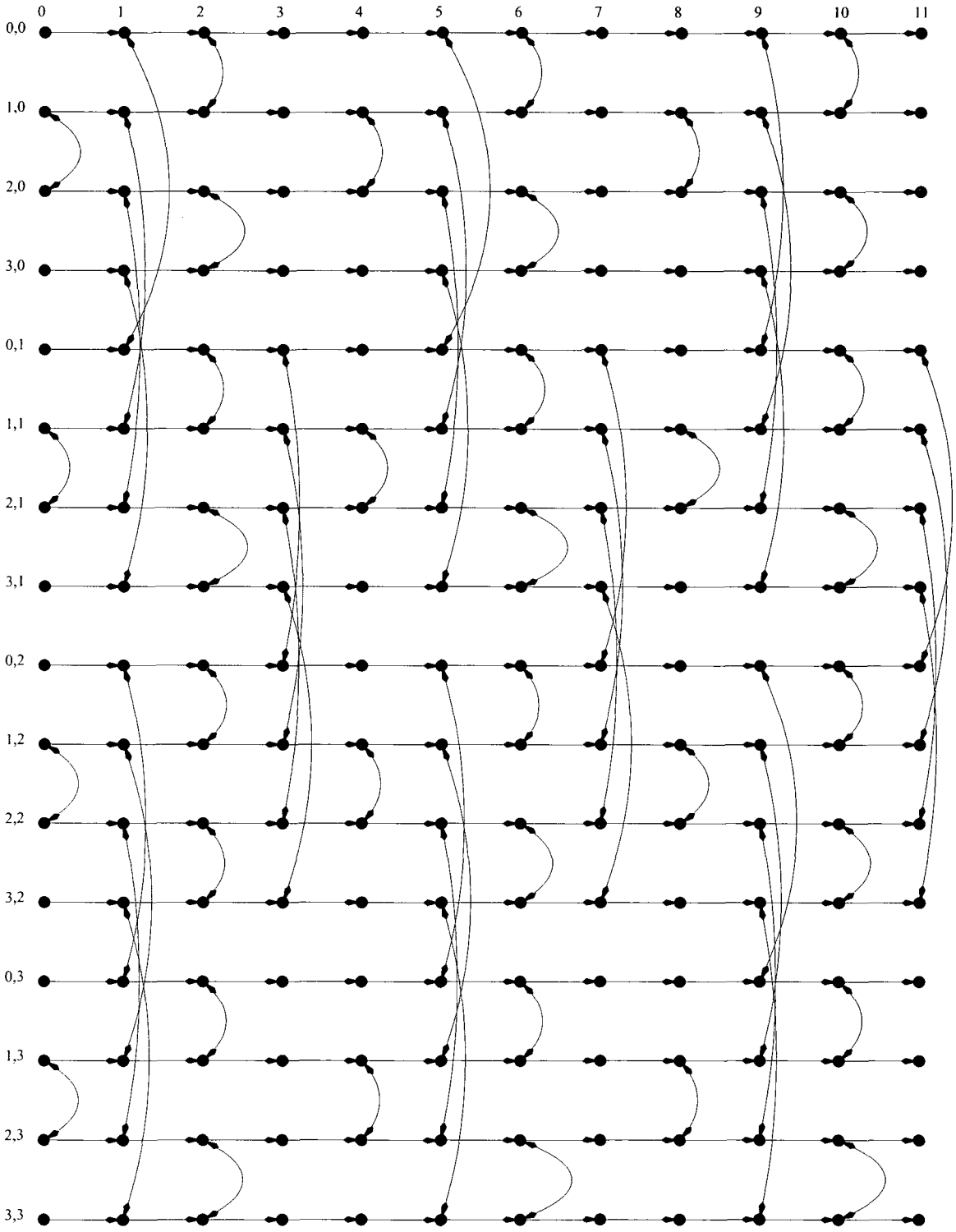


Figure 8-2: A 12-step time evolution graph of the tour network depicted in Figure 8-1

i). A path that is in the cone of (a, i) and the funnel of layer b (slice j) is called a path from (a, i) to layer b (slice j). A path from (a, i) to a layer b is called an *earliest path* if no other such path is in the funnel of an earlier slice. If a path p from (a, i) to layer b ends at (b, j), then $j - i$ is called the *delay from* (a, i) *to layer* b *along* p. The delay along the earliest path is called the delay between (a, i) and layer b.

The notion of an earliest path in mobile sensor networks plays a role analogous to that of the shortest path in static networks: they both dictate the rate of information dissemination among network nodes. The delay from a vertex (a, i) to a layer b depends not only on the way the tours are connected, but also on the design of each tour itself. Figure 8-3 shows the time evolution graphs of two different tour networks on four sensor nodes. The left graph is a 5-step time evolution graph of the circle tour (from Figure 8-1), but applied to a collection of four sensor nodes. The right graph is the time evolution graph of a new tour network, which we call the *cross* tour: in this case, the order in which a sensor node attends its south and west meetings is reversed. The effect of this change is that the delay from $((0, 0), 0)$ to layer $(1, 0)$ is increased from two to three. Thus, delay, an important performance measure of a tour network, is sensitive to the tour chosen to build the network.

The impact of a particular tour on delay should be an important consideration when designing a tour network. Delay, however, captures only part of the cost of a meeting. In static networks, the capacity of a link on a path and the load on this link also plays a role in determining end-to-end latency. A low capacity link that is congested can become a bottleneck and increase latency of the traffic that is forced to cross it. In tour networks, the relevant link capacity is the rate of data transfer during a meeting, and the relevant load is the amount of data scheduled for transmission during a meeting. Because mobile sensor networks are usually deployed in environments with no communication infrastructure external to the mobile sensor

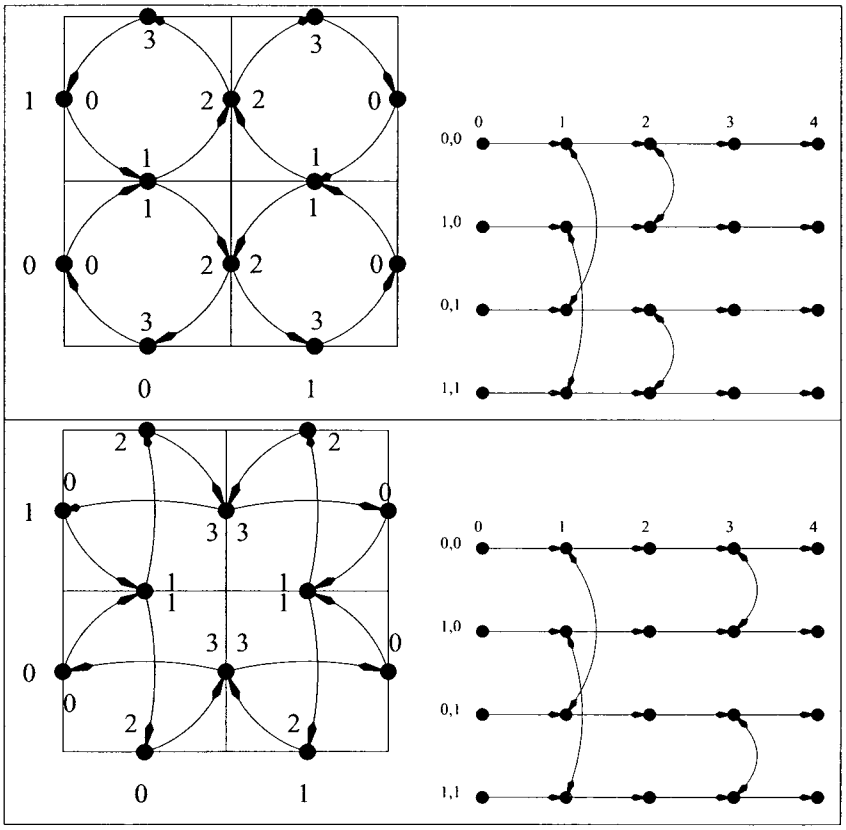


Figure 8-3: Tour networks and time evolution graphs of two tour networks differing only in the order in which meeting points are scheduled in the tour of each agent.

nodes themselves (e.g., underwater sensing), and because of their sparseness, this data rate is typically small. in contrast, sensors may produce data that takes a nontrivial amount of time to transmit across such a link. As a result, the duration of a meeting may be lengthened and may vary. A mobile sensor node spends the duration of its mission cycle sensing (traveling), computing, and meeting (possibly traveling and communicating). The sensing rate of a mission is defined as the number of samples to be gathered per unit time by a sensor node. Thus, missions requiring a high sensing rate produce a large amount of data in a short period of time. In such cases, the meeting time may be a significant portion of the total cycle time. Hence, it is important to consider the capacity of meetings in a tour network and their impact on the tour network design.

8.2 Capacity-constrained binary tour networks

We call a meeting *binary* if it involves exactly two participants, and a tour network in which all meetings are binary, a *binary tour network*. We consider binary tour networks in which the capacity of *every arc* in the time evolution graph of the tour network is fixed and set to one. Observe that this implies two constraints:

1. Each participant of a (binary) meeting is allowed the transfer of at most one segment of data.

2. Each sensor node is allowed to store and carry at most one segment of data (excluding the data that it may be currently sensing and gathering).

The notion of a segment is free to be defined by the designers: it may be a fixed number of bytes, or a single data packet, or a fixed number of data packets.

The unit capacity constraint has important implications on the design of the tour network. For tour networks with unconstrained capacity, it is clear that as long as the

time evolution graph of the tour network is connected, information may be transferred between any pair of sensor nodes with delay equal to the number of arcs traversed by the data. With unit capacity tour networks, mere connectivity is no longer sufficient to minimize delay. For example, in the unconstrained circle tour network of Figure 8-3, data from any source sensor node can be relayed to any other destination sensor node within a single cycle. When the arcs of its time evolution graph are constrained to unit capacity, some transport demands cannot be met within a single cycle: If $(0,0)$ wishes to send data to $(1,1)$, it must first send it to $(0,1)$. Because of the unit capacity constraint on the arc between slice 1 and 2 on layer $(0,1)$, $(0,1)$ is forced to send its own data from the previous cycle to $(0,0)$. Now, if $(0,1)$ wishes to send its data to $(1,1)$ it cannot do so, because its data resides with $(0,0)$, which has no sequence of meetings in the current cycle through which it could reach $(1,1)$.

Indeed, this forced exchange of data is a direct implication of the unit capacity constraint on layer arcs: if a binary meeting involves a transmission of a segment, then it must be an exchange of segments.

8.3 Unit capacity tour networks as switching networks

Because meetings are exchanges and every sensor node is responsible for carrying a single segment of data in unit capacity binary tour networks, their time evolution graphs can be viewed as switching networks. We explore the implications of this observation below.

A 2×2 switch is a four-state device with two input and two output ports, which we label as x_0, x_1 and y_0, y_1 respectively. In the first state—we call it the *pass* state—the switch routes data from x_i to y_i. In the second state, which we call the *cross* state, the switch routes data from x_i to $y_{i+1 \bmod 2}$. (In the remaining two states, which we call *copy-down* and *copy-up*, the switch copies x_0 or x_1 to both outputs respectively.

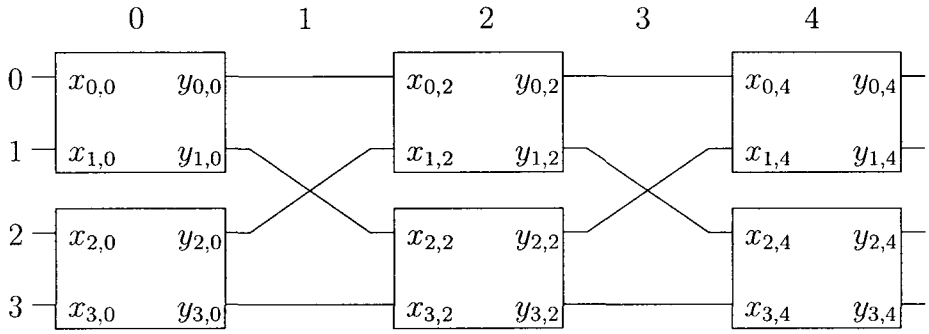


Figure 8-4: An example of a three-stage switching network.

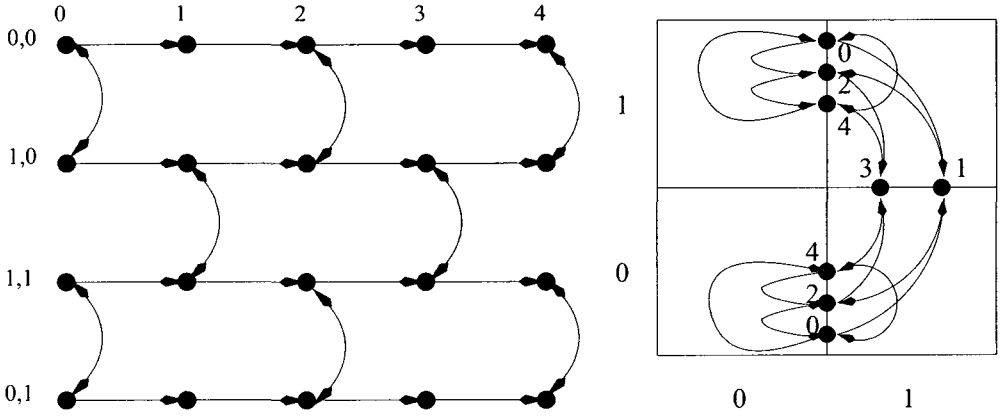


Figure 8-5: The switching network of Figure 8-4 translated into a time evolution graph of a tour network.

We shall exclude these states from the present discussion.) The state of a switch can either be set externally or, in the case of so called self-routing switching networks, using information contained in the header of each data packet to be routed.

Switches can be connected together to form switching networks. Figure 8-4 shows an example of a three stage switching network containing six switches known as a Beneš network [20]. The rectangles represent 2×2 switches and the lines represent wires connecting them. The network has four inputs and four outputs. We label inputs from 0 to 3. We shall borrow the terminology of slices and layers from tour networks and label every slice of switches as well as every slice of wiring between switches. Thus, we label five slices in our current example. We refer to input or

output points in the switching network using labels $x_{i,j}$ and $y_{i,j}$ respectively, when we wish to refer to the input point in layer i and slice j.

Figure 8-5 (left) shows how the switching network of Figure 8-4 can be interpreted as a time evolution graph of a tour network and the corresponding tour network itself is shown on the right. The translation is as follows. Every input represents a sensor node. Every intersection of wires including those within a switch translates into a binary meeting. Thus, mobile sensor nodes can be thought of as implementing the "wiring" of a switching network through their individual tours.

The tour network depicted in Figure 8-5 inherits desirable properties from the switching network that it realizes. Any permutation of transport demands can be routed by this tour network within a single cycle while respecting capacity constraints. The tour network provides a level of intrinsic tolerance of failed meetings. Moreover, this fault-tolerance can be amplified using more sophisticated switching network schemes. For a large class of permutations, routing decisions can be made locally at each meeting. The Beneš switching network can be recursively extended to larger number n of sensor nodes, as long as n is a power of 2. Correspondingly, the Beneš tour network can also be extended, while retaining the locality of meetings. The recursive structure of the Beneš switching network could also be exploited to obtain a hierarchical tour network where "local" transport demands are met within a single cycle whereas long-distance demands may take several cycles or a single super-cycle.

8.4 Tour-network transformations

We envision the tour network to be designed offline for a computational mission. As the mission progresses and as more information becomes known, the sensor nodes may find it necessary to change their pattern of communication. In such a situation, the sensor nodes may decide to change the topology of the tour network. Failure of sensor

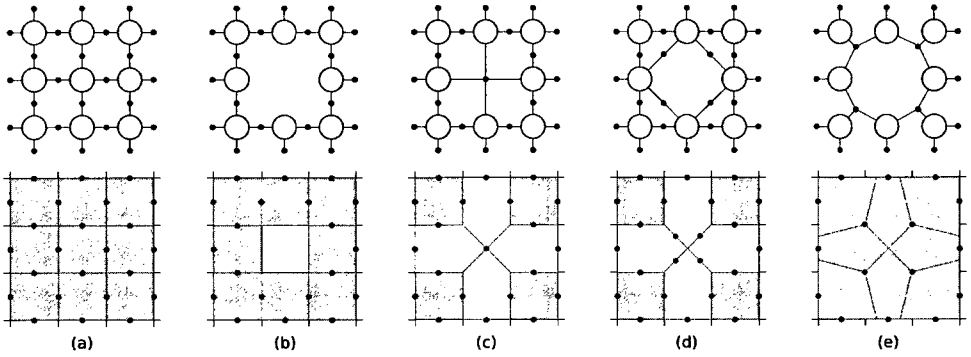


Figure 8-6: An example of topological transformations after a node failure and corresponding geometric transformations

nodes, absent nodes scheduled to participate in a meeting, or emerging communication constraints are some examples that could trigger such a network transformation.

When sensor nodes participate in a meeting, they could decide to move the location and schedule of the meeting point for the next cycle if they jointly decide that the new time and location will result in better trajectories. In contrast to such transformations involving only meeting times and tour area geometry, more elaborate transformations can modify the tour network topology by adding, removing or merging meeting points, or by modifying the membership of one or more meetings.

Such topological and geometrical transformations may be interdependent: A topological transformation may trigger geometrical adjustments and vice-versa. For instance, a topological step that removes a meeting point from a network can be followed by geometrical steps that adjust tour areas and trajectories now that there is no need for nodes to visit the meeting point that was eliminated. Conversely, changes in tour areas and in the locations of their meeting points may result in a suboptimal topology in which geometrically close sensor nodes do not interact directly, thus triggering a topological step to create a new meeting point.

To illustrate the changes that take place in the tour transformation layer, the external events that trigger them and the mission-related metrics that guide them,

consider the following scenario. A regular grid of tours is deployed for a sensing mission during which, at one point, a sensor node is permanently lost. Fig. 8-6 represents a possible series of steps that other nodes could take to adapt the network after such a failure. The top row outlines the topological transformations undertaken by the network; the bottom row describes the evolution of tour areas, meeting point locations and average scanning rates. Each tour area is drawn in the bottom part of the picture, using gray levels that represent scanning rates (the darker the gray, the higher the rate).

In (a), the network is in its stable state: each sensor node monitors a unit square using a trajectory that visits four meeting points per cycle (up, down, left and right) and the entire area is sampled with high rate. In (b), the sensor node in charge of the central tour fails. The corresponding area is then not monitored at all (null sensing rate). In (c), the four neighbors of the missing sensor node detect (at different times) that the node is missing, based on the fact that it is not attending its scheduled meetings. They unilaterally switch to a new agreed upon pattern in which the lost node is replaced with a new meeting point attended by all of its four neighbors. At the same time, the tour area of the missing sensor node is split into four triangles used by all four neighbors to extend their own tours. Since the tour areas of these sensor nodes get larger, they are rendered using a lighter gray to represent the decrease in scanning rate. So far, the sensor nodes in charge of the four corner tours have not been involved. They continue to scan their areas with the same rate and they attend the same meetings as before. In stage (d), the four nodes already involved in the operation decide to split the new meeting point into four pairwise meetings, presumably because a meeting of four nodes may have been deemed undesirable by the mission planner. Finally, in (e), the sensor nodes responsible for the four corner tours are notified of the failure as asked to adjust to the new network. Four sets of three pairwise meetings

are merged into four ternary meetings and tour area boundaries and meeting point locations are recalculated to balance the sensing workload. This results in a new network fragment that is uniformly scanned with a sensing rate not quite as high as what it is in the rest of the network, since an area originally monitored by nine sensor nodes is now handled by eight. Moreover, each sensor node now attends three meetings, presumably because the mission planner regarded this as more desirable than having some nodes attend five meetings.

For each stage of the scenario above, the tour transformation layer implements a simple distributed protocol that guarantees that the transformation is successfully implemented. The first operation (replacement of a missing agent with a meeting point) uses no communication at all among the participants, but requires an agreed-upon location and time for the new meeting. The second operation can be implemented in a centralized way, since all four nodes attend a meeting in which they all participate. The last operation, however, requires a true distributed protocol because the three nodes involved form a clique of pairwise meetings but there is no single meeting of all three nodes.

8.5 Summary and future work

In this chapter, we outlined a scheme for choreographing communication in a sparse collection of mobile sensors. The scheme proposes the abstraction of a tour for each mobile sensor node and a schedule of points at which the mobile sensors can meet and interact with each other. We provided some examples of such tour networks and showed how these could be viewed as switching networks. We also outlined how such tour networks could be transformed to react to failures and mission constraints.

Viewing tour networks as switching networks suggests a natural way of thinking about routing in tour networks. For unit-capacity networks, permutation routing

results for switching networks apply directly. Thus, self-routing switching networks induce a routing scheme for such tour networks. This needs further exploration. A simple protocol for setting up a permutation and rearranging previous paths is required. The protocol may benefit from viewing switch states (e.g., cross or pass) as tour network transformations. In any case, the relationship between transformations and switching networks merits further study. The tradeoffs in designing tour networks that are constrained, but not to unit capacity, also need further study. There is also related work on designing fault tolerant switching networks. Faults in switches correspond to failed meetings in tour networks. Whether fault tolerant switching networks imply interesting fault tolerant tour networks also merits further study. In the context of VLSI chip design, the question of optimal layouts for switching networks has also been studied. In this setting, a minimal area layout is desirable. In case of tour networks, the area to be sensed is given, and the best tour network that achieves a desired communication pattern is desired. The right cost model in this setting needs further thought. Switching networks optimized for specific computational tasks should be studied for their role in the design of mission-specific tour networks.

Chapter 9

Self-similar functions and population protocols: a characterization and a comparison

Chandy et al. proposed the methodology of "self-similar algorithms" for distributed computation in dynamic environments. We further characterize the class of functions computable by such algorithms by showing that self-similarity induces an additive relationship among the level-sets of such functions. Angluin et al. introduced the population protocol model for computation in mobile sensor networks and characterized the class of predicates computable in a standard population. We define and characterize the class of self-similar predicates and show when they are computable by a population protocol.

9.1 Introduction

Mobile wireless sensor networks hold tremendous promise as a technology for sampling a variety of phenomena at unprecedented granularities of time and space. Such networks embody a modern-day "macroscope": an instrument that can potentially revolutionize science by enabling the measurement, understanding—and eventually—control, of a whole new class of physical, biological, and social processes. The source

of potential of such networks lies in the following four capabilities endowed to each participating node: the ability to sense environmental data, the ability to compute on such data, the ability to communicate with peers in the network, and the ability to move in its environment. A network of autonomous underwater vehicles (AUVs) deployed to patrol a harbor, to map the locations of underwater mines, to monitor the diffusion of a pollutant in a river, or to build a bathymetric map are some realistic examples of missions that mobile sensor networks are charged with today.

While there has been tremendous interest in building such networks in recent years, most of this work has focused on a proper subset of the four capabilities of mobile sensor nodes described above. Work on mobile ad hoc networks has focused on mobility and communication [110, 56, 31] and sensor network research has mostly focused on sensing and communication [51, 87]. More recently, there has been a growing interest in in-network computation and communication in static sensor networks [49, 50]. We believe that all this previous work paves the way for a more comprehensive model that includes all four of the above abilities, particularly computation. Such a model would allow us to frame new questions from the point of view of the *computational mission* of the network and provide us insight into the design tradeoffs of such networks for various classes of missions. This work represents an intermediate step toward this goal.

In this work, we focus on two recent papers that deal with distributed computation in dynamic environments—the first by Chandy et al. [30] and the second by Angluin et al. [12]—and attempt to characterize the relationship between their work. Both papers are motivated by the need to understand computation in distributed systems that exist in highly dynamic environments similar to those in which mobile sensor networks are deployed. Using approaches that complement each other, these papers attempt to abstract the four capabilities of mobile sensor nodes described above

to answer new questions regarding computation in such networks. Chandy et al. propose a methodology for designing algorithms for dynamic distributed systems and identify a class of functions amenable to their method. They outline a method for systematically designing such algorithms which they call "self-similar algorithms." (We call the functions computed by such algorithms self-similar functions.) The approach taken by Angluin et al. complements that of Chandy et al. in that instead of starting with a class of functions, Angluin et al. define a computational model called the population protocol model, which abstracts the four capabilities of mobile sensor nodes described above. Their model comprises a population of anonymous identical nodes, each with a small constant amount of memory, that communicate and compute opportunistically during encounters with each other. In a series of papers [12, 13, 14], Angluin et al. have characterized the class of predicates computable in a standard population model. The goal of this work is to further characterize the class of functions defined by Chandy et al. and to understand its relationship to the computational model defined by Angluin et al.

Our contributions are as follows. Restricted to a finite input space (but any number of sensing nodes), we study the structure of self-similar functions and show how their definition imposes an additive relationship on the level-sets of such functions, a property that is similar to the one known to hold for predicates computable by population protocols in a standard population. Using these results and known results about population protocols, we show that although population protocols and self-similar functions are identically motivated, these two concepts do not coincide. For a given convention of representing predicates, we define and characterize the class of self-similar predicates and those computable by a population protocol. While self-similarity more generally captures the properties required of a function to be distributedly computable in a dynamic environment, the constraints that its definition

imposes appear to be stronger than those imposed by population protocols. On the other hand, the notion of self-similarity appears to be more general than the notion of opportunistic computation in a population protocol. Our work contrasts these two conceptions of computation in dynamic environments in a mobile sensor network and highlights their particular strengths. We hope that this increased understanding of existing models will usher in better models of mobile sensor networks that incorporate the computational mission of such networks. We also hope that this work will generate interest in a study of mobile sensor networks that unifies computation, communication, mobility, and sensing.

9.2 Self-similar algorithms

Implicit in the paper by Chandy et al. [30] are the following questions: How can we derive distributed algorithms that compute correctly in dynamic environments? Which functions are amenable to distributed computation in dynamic environments? To answer the first question, Chandy et al. begin by enumerating properties that a computation must possess, if it is to execute correctly in a dynamic distributed environment. They restrict their investigation to stable and idempotent functions which can be computed by what they call "self-similar algorithms." By stability, it is meant that once a computation achieves its "final" state, it remains in that state forever, thus providing a stable answer. It follows that a computation in such an environment must be conservative in the sense that it must always transition to only those states that would not result in an incorrect computation; all transitions must conserve the correct final answer. That is, if s_i is the collective state of the computational agents in the system in the ith step and f is the function to be computed, then $f(s_i) = f(s_0)$ for all i. Finally, a self-similar algorithm is one in which any "group behaves like the entire system" [30]. More precisely, suppose f is a function that is to be computed

by a collection of agents. Then, a self-similar algorithm A for f is one which can be executed by any (nonempty) subset of identical agents participating in a sequence of arbitrary groupings such that the result of their "local" computation is compatible with and usually contributes to the "global" computation that is to be executed. Chandy et al. show that the above properties—stability, idempotence, conservation, and computability by self-similar algorithms—hold exactly for a class of functions they call *superidempotent.*

Definition 1. *A function f from multisets to multisets is* **superidempotent** *if $f(X \cup Y) = f(f(X) \cup Y)$ [30].*

In this paper, we shall refer to such functions as self-similar functions to emphasize their computability by a self-similar algorithm.

9.2.1 General observations

It is easy to see that the class of self-similar functions excludes some familiar functions.

Proposition 1. *Any one-to-one function (except for the identity) is not self-similar, because it is not idempotent.*

On the other hand, self-similar functions include some familiar functions.

Proposition 2. *An idempotent homomorphism is self-similar.*

Proof. If f is an idempotent homomorphism, then the r.h.s. in the definition of superidempotence $f(f(X) \cup Y) = f(f(X)) \cup f(Y)$ (by homomorphism), $= f(X) \cup f(Y)$ (by idempotence) $= f(X \cup Y)$ (by homomorphism), which is the l.h.s. of the definition of superidempotence. □

Corollary 1. *Let T be a linear transformation such that $T^2 = T$. Then, T is self-similar.*

Proof. By its definition, T is idempotent and linearity implies the homomorphism property. □

Thus, all projections (i.e., linear transformations T such that $T^2 = T$) are self-similarly computable.

9.2.2 Finite-valued self-similar functions

While Chandy et al. define self-similar algorithms over infinite input spaces, in order to compare the class of such functions with a realizable model of computation, we study self-similar functions over a finite input space (alphabet). Let Q be the finite nonempty set of possible input values for any agent and let $|Q| = q$ be a positive integer. Consider the q-dimensional space $\mathbb{N}^q$ of nonnegative integers.

Lemma 1. *Let $Q^\star$ be the (infinite) set of all finite multisets containing elements from Q. There exists an isomorphism ϕ between the monoids $(Q^\star, \cup)$ and $(\mathbb{N}^q, +)$.*

Proof. Any multiset $S \in Q^\star$ can be written as $\{s_1^{(m_1)}, s_2^{(m_2)}, \ldots, s_q^{(m_q)}\}$ where m_i denotes the nonnegative integer number of times that s_i appears in S. Define $\phi : Q^\star \to \mathbb{N}^q$ as $\phi(S) = (m_1, m_2, \ldots, m_q)$. It is easy to verify that ϕ so defined is a bijective homomorphism. □

Thus, the definition of superidempotence can be translated from $(Q^\star, \cup)$ to $(\mathbb{N}^q, +)$ as follows: a function $f : \mathbb{N}^q \to \mathbb{N}^q$ is superidempotent if and only if $f(x + y) = f(f(x) + y)$ for all $x, y \in \mathbb{N}^q$. In this subsection, $f : \mathbb{N}^q \to \mathbb{N}^q$ is a self-similarly computable function. From the definition of superidempotence, we know

Fact 1. *For any $u, v \in \mathbb{N}^q$, $f(u + v) = f(f(u) + v) = f(u + f(v)) = f(f(u) + f(v))$.*

Definition 2. *For any $v \in \mathbb{N}^q$, denote by $\sum v$ the integer $\sum_{i=1}^{q} v_i$, and by H_k^q the hyperplane $\{v \in \mathbb{N}^q | \sum v = k\}$.*

We assume that a computational step is agent conserving in that the number of a agents in a group participating in a computational step does not change during the step. From this we have

Fact 2. *If* $f : \mathbb{N}^q \to \mathbb{N}^q$ *and* $v \in \mathbb{N}^q$, $\sum v = \sum f(v)$.

That is, any $v \in \mathbb{N}^q$ lives in the $q-1$-dimensional hyperplane $\{u : \sum u = \sum v\}$ of $\mathbb{N}^q$ and any self-similar f maps v to an $f(v)$ in the same plane. It is useful to know the number of points in each H_k^q. For each $k \in \mathbb{N}$, the number of points in H_k^q is the number of integral solutions of the equation $\sum v = k$ Therefore, $|H_k^q| = \binom{k+q-1}{q-1}$.

Definition 3. *The set of all points (multisets)* x *such that* $f(x) = y$, *for some fixed* y, *is called a* **fiber**. *A fiber is* **trivial** *if it contains exactly one point. Any subset of a fiber of* f *is called a* **contour of** f. *A contour of* f *containing* u *that also contains* $f(u)$ *is called a* **complete contour.** *The* **value** *of a contour is* $f(u)$ *for any* u *in the contour.*

Self-similar computations progress along trajectories that must be contained in fibers; if not then f cannot be conservative. Fibers play a central role in self-similar functions. Indeed, self-similarity induces an additive relationship between contours, as we show below.

Theorem 1 (Direct sum of contours). *If* U *and* V *are contours, then* $U \oplus V = \{u + v | u \in U, v \in V\}$ *is also a contour.*

Proof. Since the claim is trivially true if either U or V are empty, we assume that they are both nonempty. For any $w_1, w_2 \in U \oplus V$ let $w_1 = u_1 + v_1$ and $w_2 = u_2 + v_2$ for some $u_1, u_2 \in U$ and $v_1, v_2 \in V$. Now $f(w_1) = f(u_1 + v_1) = f(f(u_1) + f(v_1)) = f(f(u_2) + f(v_2)) = f(u_2 + v_2) = f(w_2)$, where the second and fourth equalities follow from the definition of superidempotence, and the third from the definition of a contour. □

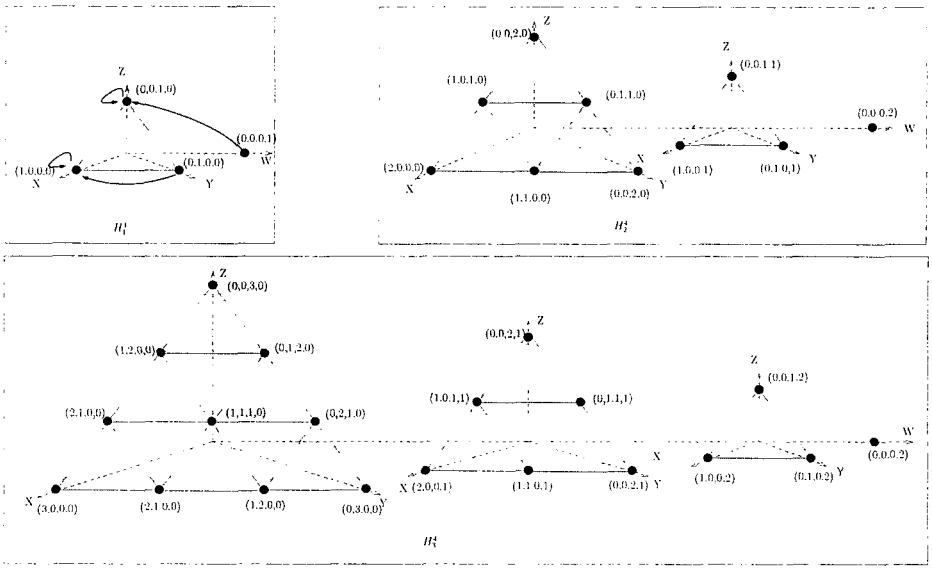


Figure 9-1: Relationship between contours of a self-similar function $f : \mathbb{N}^4 \rightarrow \mathbb{N}^4$. (The axes in order are X, Y, Z, and W.) Points (black disks) included in the same shaded region form a contour. Notice that H_2^4 contains four copies of H_1^4, and H_3^4 contains four copies of H_2^4. Contours are invariant under translation from H_k^4 to H_{k+1}^4.

Corollary 2 (Translation of a contour). *If U is a contour, then for any $v \in \mathbb{N}^q$, the translation $U + v = \{u + v | u \in U\}$ of U is also a contour.*

Proof. For any $v \in \mathbb{N}^q$, $\{v\}$ is (trivially) a contour. Then by Theorem 1, $U \oplus \{v\}$ is a contour. □

Figure 9-1 illustrates this relationship between contours of a self-similar function $f : \mathbb{N}^4 \rightarrow \mathbb{N}^4$. Over H_1^4, it is defined as follows: $f(0,1,0,0) = f(1,0,0,0) = (1,0,0,0)$ and $f(0,0,0,1) = f(0,0,1,0) = (0,0,1,0)$. Thus, there are two fibers (and contours) in H_1^4: $\{(0,1,0,0),(1,0,0,0)\}$ and $\{(0,0,0,1),\ (0,0,1,0)\}$. (There are only $\binom{1+4-1}{4-1} = 4$ points in H_1^4 and they are thus partitioned into two fibers.) As per the above results, any translation of these two contours must also be a contour. Thus, $\{(0,0,0,1) + (1,0,0,0),(0,0,1,0) + (1,0,0,0)\} = \{(1,0,0,1),(1,0,1,0)\}$ for example, must be a contour in H_2^4. (There are three more possible translations,

one along each of the axes, all of which must also be contours in H_2^4.) Since points in each contour must have the same value, the union of intersecting contours must form a single contour. For example, the intersecting contours $\{(1,0,1,0),(1,0,0,1)\}$, $\{(0,1,1,0),(0,1,0,1)\}$, $\{(1,0,1,0),(0,1,1,0)\}$, and $\{(1,0,0,1),(0,1,0,1)\}$ of H_2^4 together form a single contour, as shown by the overlapping shaded regions of the figure. Similarly, the contours in H_2^4, when translated along any of the four axes must form contours in H_3^4, as shown in the figure.

Viewing contours in translation justifies naming such functions as self-similar: contours in H_k^q are the result of translating contours in H_{k-1}^q in q ways and are thus copies of them; those in H_{k-1}^q are copies of those in H_{k-2}^q; and so on. However, while contours are invariant under translation, the value of a contour in H_k^q, in contrast to the standard notion of self-similarity, need not bear any relationship to the value of a contour in H_{k-1}^q. For example in Figure 9-1, the value of any point in H_2^4 under f is not determined by its contour membership: the contour only requires that the value of all its points be the same.

The results proved above are fundamental in understanding the structure of self-similar functions. They complement the description given by Chandy et al. that self-similar algorithms are those in which "any group behaves like the entire system" [30]. Our results show that for finite input spaces, such algorithms compute functions in which self-similarity manifests itself in the form of an additive relationship between contours: larger contours are formed by translating smaller contours. This clarifies the notion of self-similarity proposed by Chandy et al. and makes our understanding of it more precise.

We now state two useful results that immediately follow from the above results.

Definition 4. *For any* $u = (u_1, \ldots, u_q) \in \mathbb{N}^q$ *and* $v = (v_1, \ldots, v_q) \in \mathbb{N}^q$, *we define the* **partial order** $\leq$ *as follows:* $u \leq v \iff \forall i \in \{1, \ldots, q\} : u_i \leq v_i$.

Lemma 2. *Let $\{v^1, \ldots, v^r\}$ be a contour in H_k^q (with k such that $1 \leq r \leq |H_k^q|$) and let $u \in \mathbb{N}^q$ such that $u \leq v^i$ for $i = 1, \ldots, r$. Then the set $\{u + \sum_{i=1}^r m_i(v^i - u) : m_i \in \mathbb{N}, \sum_{i=1}^r m_i = m\}$ is a contour in $H_{j+m(k-j)}^q$, where $j = \sum u$.*

Proof. We prove this by induction on r.

Basis. If $r = 1$, then we must show that if $\{v^1\}$ is a contour in H_k^q and $u \leq v^1$, then the set $\{u + m_1(v^1 - u) : m_1 \in \mathbb{N}, m_1 = m\}$ is a contour in $H_{j+m(k-j)}$, where $j = \sum u$. Since $v^1 \in H_k^q$, and $u \in H_j^q$, $u + m(v^1 - u) \in H_{j+m(k-j)}^q$. For any $m_1 = m \in \mathbb{N}$ the set in question contains a single vector and is therefore trivially a contour.

Induction hypothesis. Suppose the statement is true for $r = n$.

Inductive step. For $r = n$, we are given that the set $\{u + \sum_{i=1}^n m_i(v^i - u) : m_i \in \mathbb{N}, \sum_{i=1}^n m_i = m\}$ is a contour in $H_{j+m(k-j)}^q$. Let $v^{n+1} \in H_k^q$. Let $U_n = \{u : u \leq v^i, i = 1, \ldots, n\}$ and $U_{n+1} = \{u : u \leq v^i, i = 1, \ldots, n+1\}$. Then it must be that $U_{n+1} \subseteq U_n$ because if $u \in U_{n+1}$, then it must necessarily be no larger than $v^1, \ldots, v^n$. Moreover, $(0, \ldots, 0) \in U_{n+1}$ and hence U_{n+1} is nonempty. Thus, the induction hypothesis holds for all $u \in U_{n+1}$, since it holds for U_n. Let $u' \in U_{n+1}$ with $\sum u' = j'$. Therefore, $\{u' + \sum_{i=1}^n m_i(v^i - u') : m_i \in \mathbb{N}, \sum_{i=1}^n m_i = m\}$ is a contour in $H_{j'+m(k-j')}^q$ as per the induction hypothesis.

Now, the set $\{m_{n+1}(v^{n+1} - u')\}$ is a contour in $H_{m_{n+1}(k-j')}^q$ for any fixed $m_{n+1} \in \mathbb{N}$ because it contains a single point. Therefore, by Theorem 1, $\{u' + \sum_{i=1}^n m_i(v^i - u') : m_i \in \mathbb{N}, \sum_{i=1}^n m_i = m\} \oplus \{m_{n+1}(v^{n+1} - u')\}$ is a contour in $H_{j'+m'(k-j')}^q$, where $m' = m + m_{n+1}$. But $\{u' + \sum_{i=1}^n m_i(v^i - u') : m_i \in \mathbb{N}, \sum_{i=1}^n m_i = m\} \oplus \{m_{n+1}(v^{n+1} - u')\} = \{u' + \sum_{i=1}^{n+1} m_i(v^i - u') : m_i \in \mathbb{N}, \sum_{i=1}^{n+1} m_i = m'\}$. Thus, we have shown that this set is a contour in $H_{j'+m'(k-j')}$, where $j' = \sum u'$. □

The union of the contours described in the above result is called a *linear* set. Such sets are closely related to the type of predicates computable in the standard population protocol model.

We now state a useful special case of the above result. We omit the proof, which follows directly from the previous result, due to space restrictions.

Corollary 3. *Let $u, v^1, v^2, \ldots, v^q \in \mathbb{N}^q$ be such that $v^i = u + e_i$ where $\{e_1, \ldots, e_q\}$ is the standard basis for $\mathbb{N}^q$. If $\{v^1, \ldots, v^q\}$ is a contour, then f is constant in each H_k^q for all points $w \geq u$.*

Lemma 3. *Any function $f : \mathbb{N}^q \to \mathbb{N}^q$ that is constant over each H_k^q and maps each H_k^q to itself is self-similar.*

Theorem 2. *There exists a function $f : \mathbb{N}^2 \to \mathbb{N}^2$ that is not computable but is self-similar.*

Proof. Let $w = w_2, w_3, \ldots$ be an infinite sequence of nonnegative integers such that $0 \leq w_k < |H_k^2|$. Let $f_w : \mathbb{N}^2 \to \mathbb{N}^2$ be a function constant over each H_k^2 such that for any $(i, k-i) \in H_k^2$, $f_w(i, k-i) = (w_k, k - w_k)$: w_k defines the value of the function in H_k^2. If $w \neq w'$ are two sequences as defined above such that $w_k \neq w'_k$, then $f_w(i, k-i) = (w_i, k - w_i) \neq (w'_i, k - w'_i) = f_{w'}(i, k-i)$. Thus, every sequence w defines a distinct function f_w that is constant over each H_k^2. By Lemma 3, each such f_w is self-similar. However, the set $\{f_w | w = w_2, w_3, \ldots; 0 \leq w_k < |H_k^2|\}$ is uncountable, whereas the set of Turing machines is countable. □

9.3 Population protocols and self-similar functions

In a series of recent papers, Angluin et al. have defined the *population protocol* model of distributed computation and characterized its computational power [12]. A *population* is a collection of n anonymous computational agents with an undirected population graph on n vertices. Each agent is modeled as a deterministic finite automaton, with a finite set of transition rules from pairs of states to pairs of states. In the randomized variant (see [14] for details), an input symbol from an input alphabet

is provided to each agent, and a fixed input function maps it to the initial state of the agent. A computation evolves in discrete steps, and at each step, an edge (i, j) of the population graph is chosen uniformly at random by the "environment": this models a pairwise random *encounter* between agents during which they communicate and compute. During such an encounter, agents i and j transition from their current states q_i and q_j to new states according to the population protocol (i.e., $(q_i, q_j) \to (q_i', q_j')$). The collective state of all n agents can be completely described by an n-dimensional vector over the states of the protocol, where the ith component is the current state of the ith agent. Thus, an execution is an *infinite* sequence of n-dimensional vectors. At any step, the current output of the computation can be obtained by mapping the current state of any agent to the output alphabet using a given fixed output function. A function f is stably computed by a population protocol iff for any input assignment x, the computation eventually converges to an orbit of n-vectors, all of which map to the unique $f(x)$ under the output function. We recall some definitions below [12].

Definition 5. *A* **population protocol** $\mathcal{A}$ *is a 6-tuple* $\mathcal{A} = (X, Y, Q, I, O, \delta)$ *where:* X *is the* **input alphabet**, Y *is the* **output alphabet**, Q *is a set of* **states**, $I : X \to Q$ *is the* **input function**, $O : Q \to Y$ *is the* **output function**, *and* $\delta : Q \times Q \to Q \times Q$ *is the* **transition function**.

Definition 6. *A* **population** $\mathcal{P}$ *is a set* A *of* n *agents with a directed graph over the elements of* A *as vertices and edges* $E \subseteq A \times A$. *The* **standard population** $\mathcal{P}_n$ *is the set of* n *agents* $A_n = \{a_1, \ldots, a_n\}$ *with the complete directed graph (without loops) on* A_n.

Definition 7. *A* **semilinear set** *is a subset of* $\mathbb{N}^q$ *that is a finite union of linear sets of the form* $\{u + k_1 v_1 + k_2 v_2 + \cdots + k_m v_m\}$ *where* u *is a* q*-dimensional base vector,* $v_1, \ldots, v_m$ *are* q*-dimensional basis vectors, and* $k_1, \ldots, k_m \in \mathbb{N}$. *A* **semilinear predicate** *is one that is true precisely on a semilinear set.*

The computational power of population protocols was characterized by Angluin et al. [13, 14].

Theorem 3 (Theorem 6 in [14]). *A predicate is computable by a population protocol in a standard population if and only if the set of points on which it is true is semilinear.*

9.3.1 Self-similar functions computed by population protocols

Theorem 4. *If the population protocol* $\mathcal{A} = (X, X, Q, I, I^{-1}, \delta)$ *stably computes a function* $f : X^\star \to X^\star$ *from multisets to multisets over* X *in the standard population* $\mathcal{P}_n$*, then* f *is self-similar.*

Proof sketch. A population protocol $\mathcal{A}$ that correctly executes in a standard population $\mathcal{P}_n$ must also correctly execute in any population $P \subseteq \mathcal{P}_n$ because it cannot distinguish between the two populations. Partition $\mathcal{P}_n$ into P and P'. Let t be larger than the number of steps required for $\mathcal{A}$ to converge when executed in P and P'. Let $f(P)$ and $f(P')$ denote the output respectively. Now execute $\mathcal{A}$ in $\mathcal{P}_n$ such that for the first t steps no inter-partition encounter is allowed, and after t steps all encounters are allowed. The intermediate output will be $f(P) \cup f(P')$ and the final output will be $f(f(P) \cup f(P')) = f(\mathcal{P}_n)$.

9.3.2 Predicates: semilinear and self-similar

Definition 8. *A* **predicate** *is a function* $P : \mathbb{N}^q \to \{T, F\}$*. For any predicate* P*, its* **consensus predicate** *form is a function* $f : \mathbb{N}^q \to \mathbb{N}^q$ *such that for any* $v \in \mathbb{N}^q$*,* $f(v) = (\sum v, 0, 0, \ldots, 0)$ *iff* $P(v) = T$ *and* $f(v) = (0, \sum v, 0, \ldots, 0)$ *iff* $P(v) = F$*. We call the consensus predicate form* **self-similar** *if* f *is self-similar.*

The consensus predicate defined above follows the "all-agents output convention" as defined by Angluin [12] which requires all agents to agree on the truth-value of the predicate. In the sequel, our results involve only those predicates that are expressible

in this convention because this is one of the conventions used by Angluin et al. and we are interested in comparing self-similar predicates to population protocol computable predicates. We postpone the discussion of more robust conventions to future work.

Proposition 3. *Not all semilinear consensus predicates are self-similar consensus predicates.*

Proof. Consider the following consensus predicate: $f(i,j) = (i+j, 0)$ if $j \leq i$ and $f(i,j) = (0, i+j)$ otherwise. It is easy to show that this predicate is semilinear and idempotent but not self-similar. □

If a predicate P is always true or always false, then its consensus form function will be constant over each H_k^q, and by Lemma 3, will be self-similar. We say that a predicate is eventually constant if there is a $k \in \mathbb{N}$ such that the predicate is constant over H_k^q. (Corollary 3 implies that the predicate is then constant for all $k' \in \mathbb{N}$ such that $k' \geq k$.)

Theorem 5. *A predicate $P : \mathbb{N}^q \to \{T, F\}$ that is not eventually constant has a self-similar consensus form $f : \mathbb{N}^q \to \mathbb{N}^q$ if f is idempotent and either the set of points on which P is true or that on which P is false has a standard basis.*

Proof. Suppose the set of true points of P has a standard basis T_1^q. Thus, P is true only on points in $\mathsf{span}(T_1^q)$ and hence P is false only on points in $\mathsf{span}(F_1^q \cup (F_1^q \oplus T_1^q))$.

To show that P has a self-similar consensus form f, we must show that $\forall v \in \mathbb{N}^q : \forall u \leq v : f(v) = f(f(u) + f(v-u))$. Suppose $P(v) = T$, that is $v \in \mathsf{span}(T_1^q)$. Then $\forall u \leq v : u \in \mathsf{span}(T_1^q)$ because u must have zeroes in at least those coordinates in which v has zeroes. Now since $P(v) = T$, $f(v) = (\sum v, 0, \ldots, 0)$ by definition of the consensus form. On the other hand, $f(f(u) + f(v-u)) = f((\sum u, 0, \ldots, 0) + (\sum (v-u), 0, \ldots, 0)) = f(\sum v, 0, \ldots, 0)$. Since $f(v) = (\sum v, 0, \ldots, 0)$, and since f

is idempotent, $f(\sum v, 0, \ldots, 0) = (\sum v, 0, \ldots, 0)$. Thus, we have shown that $\forall v \in \mathsf{span}(T_1^q) : \forall u \leq v : f(v) = f(f(u) + f(v-u))$.

Now suppose $P(v) = F$. Thus $v \notin \mathsf{span}(T_1^q)$, that is $v \in \mathsf{span}(F_1^q \cup (F_1^q \oplus T_1^q)) = \mathsf{span}(F_1^q) \cup \mathsf{span}(F_1^q \oplus T_1^q)$. If $v \in \mathsf{span}(F_1^q)$, then the same argument as above applies because $\forall u \leq v : P(u) = F$.

If $v \in \mathsf{span}(F_1^q \oplus T_1^q)$, then $v = v_F + v_T$ for some $v_F \in \mathsf{span}(F_1^q)$ and some $v_T \in \mathsf{span}(T_1^q)$. Thus $P(v_F) = F$ and $P(v_T) = T$ and therefore $f(v_F) = (0, \sum v_F, 0, \ldots, 0)$ and $f(v_T) = (\sum v_T, 0, \ldots, 0)$. Therefore $f(f(v_F) + f(v_T)) = f(\sum v_T, \sum v_F, 0 \ldots, 0)$. Since P is not eventually constant, $(i, 0, \ldots, 0) \in \mathsf{span}(T_1^q)$ and $(0, j, 0, \ldots, 0) \in \mathsf{span}(F_1^q)$ for all $i, j \in \mathbb{N}$. Hence $(\sum v_T, 0, \ldots, 0) \in \mathsf{span}(T_1^q)$ and $(0, \sum v_F, \ldots, 0) \in \mathsf{span}(F_1^q)$ and therefore $(\sum v_T, \sum v_F, 0 \ldots, 0) \in \mathsf{span}(T_1^q \oplus F_1^q)$. Therefore, $P(\sum v_T, \sum v_F, 0, \ldots, 0) = F$ and hence $f(\sum v_T, \sum v_F, 0 \ldots, 0) = (0, \sum v, 0, \ldots, 0)$. Thus, we have shown that $\forall v \in \mathsf{span}(F_1^q \cup (F_1^q \oplus T_1^q))$ f is self-similar. □

Theorem 6. *If $P : \mathbb{N}^q \to \{T, F\}$ is a predicate with a self-similar consensus form, then at least one of the following holds: Either the set of points on which P is true or that on which P is false has a standard basis; or P is eventually constant.*

Proof. Consider the q points in H_1^q. If P is true on all q points or false on all q points, then P is eventually constant. So, assume otherwise and let the true fiber $T_1^q \subset H_1^q$ and the false fiber $F_1^q \subset H_1^q$ partition H_1^q (with $e_1 \in T_1^q$ and $e_2 \in F_1^q$ as per the definition of the consensus form convention).

Now consider H_2^q and observe that $H_2^q = (T_1^q \oplus T_1^q) \cup (F_1^q \oplus F_1^q) \cup (T_1^q \oplus F_1^q)$. By Theorem 1, $T_1^q \oplus T_1^q$, $F_1^q \oplus F_1^q$ and $T_1^q \oplus F_1^q$ are all contours and thus P is constant over each of these sets in H_2^q.

For some $v \in T_1^q \oplus F_1^q$, suppose $P(v) = F$. Then, all points in $T_1^q \oplus F_1^q$ must map to $(0, 2, 0, \ldots, 0)$ since P must be false on all these points. Furthermore, $f(0, 2, 0, \ldots, 0) = (0, 2, 0, \ldots, 0)$ since f is self-similar and hence idempotent. But $(0, 2, 0, \ldots, 0) \in$

$F_1^q \oplus F_1^q$ and hence P must be false on all the points in $F_1^q \oplus F_1^q$. Thus, we can write the false fiber $F_2^q \supseteq (F_1^q \oplus F_1^q) \cup (F_1^q \oplus T_1^q) = F_1^q \oplus (F_1^q \cup T_1^q) = F_1^q \oplus H_1^q$. (It is easy to check that $\oplus$ distributes over $\cup$.) The only points remaining in H_2^q are those in the contour $T_1^q \oplus T_1^q$. If P is false on any of these points, then P is constant on H_2^q, and thus P is eventually constant. So assume that P is true on each point in the contour $T_1^q \oplus T_1^q$. Therefore, the set of true points in H_2^q, i.e., the true fiber in H_2^q is $T_2^q = T_1^q \oplus T_1^q$ and the false fiber $F_2^q = F_1^q \oplus H_1^q$. Thus, H_2^q is partitioned into two nonemtpy fibers.

Now $\mathbb{N}^q = \mathsf{span}(H_1^q) = \mathsf{span}(T_1^q) \cup \mathsf{span}(F_1^q) \cup \mathsf{span}(F_1^q \oplus T_1^q) = \mathsf{span}(T_1^q) \cup \mathsf{span}(F_1^q \cup (F_1^q \oplus T_1^q)) = \mathsf{span}(T_1^q) \cup \mathsf{span}((F_1^q \cup F_1^q) \oplus (F_1^q \cup T_1^q)) = \mathsf{span}(T_1^q) \cup \mathsf{span}(F_1^q \oplus H_1^q) = \mathsf{span}(T_1^q) \cup \mathsf{span}(F_2^q)$. Using Corollary 3 and considering F_2^q as the contour we obtain that the $\mathsf{span}(F_2^q) \cap H_k^q$ is a contour in every H_k^q, $k \geq 2$. If the value of this contour in any H_k^q is true, then P is constant over all of that H_k^q and thus is eventually constant. If the value of this contour is false for all H_k^q, and for some k, the value of T_k^q is also false, then P is constant over all of H_k^q and thus is eventually constant. If the value of this contour is false for all H_k^q, and the value of T_k^q is true for all H_k^q, then the set of true points has a standard basis T_1^q.

We assumed that for some $v \in T_1^q \oplus F_1^q$, $P(v) = F$. If we assume that $P(v) = T$, then we can show that the set of false points has a standard basis F_1^q. □

From this, and Angluin et al.'s Theorem 3 immediately follows

Theorem 7. *If predicate $P : \mathbb{N}^q \to \{T, F\}$ is not eventually constant and has a self-similar consensus form, then P is computable by a population protocol.*

Proof. Since P has a self-similar consensus form and is not eventually constant, the set of points on which either P or its negation is true has a standard basis and is therefore a linear set. Population protocols are closed under complement. □

For predicates that are eventually constant, self-similarity imposes no additional constraints within each H_k^q. Thus, for any $k \in \mathbb{N}$, the predicate may be true on all points in H_k^q or false on all points in H_k^q. Therefore, the computability of such predicates by a population protocol is given directly by Theorem 3.

9.3.3 Self-similar functions not computable by population protocols

It is known that all predicates that are computable in the standard population are in the class NL [12], the set of functions computable by a nondeterministic Turing machine with access to memory logarithmic in the size of the input.

Theorem 8. *There exists a self-similar function that is in NL but whose predicate form is not computable by any population protocol.*

Proof. Let $f : \mathbb{N}^2 \to \mathbb{N}^2$ be the constant function such that for any $(i, k-i) \in H_k$, $f(i, k-i) = (k - \lfloor \lg k \rfloor, \lfloor \lg k \rfloor)$. By Lemma 3, f is self-similar. Since f requires an addition, the counting of the number of bits of the result, and a subtraction, it is in L, the set of functions computable with a deterministic Turing machine with access to memory logarithmic in the size of the input. It is known that $\mathrm{L} \subseteq \mathrm{NL}$ [88] and thus f is in NL. Define the predicate $P_f(v)$ over all points $v \in \mathbb{N}^2$ such that it is true if and only if $v \in H_k^2$ is the image of all $u \in H_k^2$ under f. From Theorem 3 we can deduce that the predicate P_f will be computable by a population protocol if and only if the set of its true points—which are also the fixed points of f—is semilinear. However, the set of fixed points $\{(k - \lfloor \lg k \rfloor, \lfloor \lg k \rfloor) | k = 1, 2, \ldots\}$ of f is not semilinear. □

9.4 Conclusions and future work

Starting with the class of self-similar algorithms defined by Chandy et al., we studied functions from multisets to multisets computed by such algorithms over a (finite)

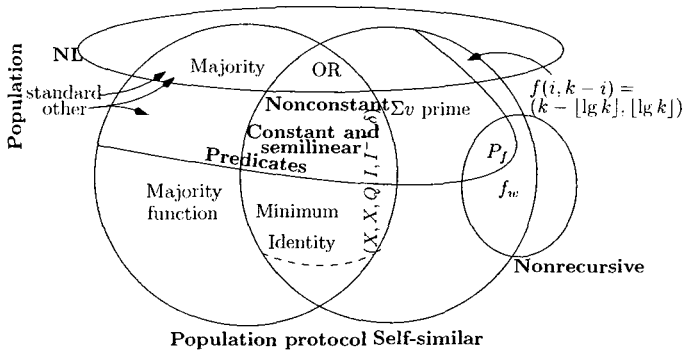


Figure 9-2: Relationship between self-similar functions and functions computable by population protocols. (Bold names differentiate classes from examples. Not all relationships are known.)

input alphabet. We showed how the definition of self-similarity of algorithms—a group of any size behaves identically—results in a self-similar additive relationship among the contours of the functions computed by such algorithms. We defined self-similar predicates under the consensus convention used by Angluin et al., and showed that all such predicates that are not eventually constant have a simple structure: the set of points on which they are true or the set of points on which they are false has a standard basis. Using known results about population protocols, we thus showed that nonconstant self-similar predicates are computable by population protocols. We also showed that the notion of self-similarity is more general than, though quite similar to, the notion of opportunistic computability inherent in the population protocol model by showing the existence of a self-similar function not computable by population protocols. Our results, along with other examples, are summarized in Figure 9-2.

Both models discussed in this work are motivated by distributed computation in dynamic mobile sensor network-like environments. However, neither model attempts to capture in sufficient detail the spatio-temporal nature of the data and its impact on communication and computation. Thus, one cannot frame questions that involve

the spatial distribution of data or constraints on communication in the context of these models. If the state space of the population protocol model is endowed with a topology reflecting the space in which the network exists, then such questions may perhaps be framed. The population protocol model is intended to model a large number of frugal sensors. This may not be appropriate for AUV networks where the number of AUVs is small and each AUV is equipped with sufficient resources. While other models may allow us to ask these questions, we believe that a unified approach to studying such networks may be necessary.

Chapter 10

Undecidability in graph products

Computing invariants in graph products has led to several challenging open questions. It is plausible that some of these invariants are uncomputable. We present undecidability results about two graph products. For the direct product, we show that the question of existence of transposable walks is undecidable. We define a lexicographic variant of the direct product. For this new product, we show that the transpose independence ratio of infinite families of graphs generated by this product is not computable. Both proofs use the undecidability of the tiling problem, but in different ways. Our definition of transposable walks and independent sets may be of general interest.

10.1 Introduction

A graph comprises a set and a relation on the set. We shall only consider finite sets and binary relations on them; thus, our discussion will be restricted to finite graphs. The elements of the relation are represented by ordered pairs. Such graphs are called directed graphs. In this case, the elements of the relation are called arcs. When the relation is symmetric, the ordered pairs in the relation can be replaced by unordered sets containing the two elements. Such graphs are called undirected graphs. In this case, the elements of the relation are called edges. If the relation is irreflexive, the the graph is said to be simple; otherwise we say that the graph contains loops.

We shall denote a graph with upper-case letters $G, H, \ldots$. The vertex set of a graph G will be denoted by $\mathcal{V}(G)$ and its arc or edge set will be denoted by $\mathcal{E}(G)$.

We are interested in investigating the product operation on graphs. Given graphs G and H, we are interested in obtaining a new graph that has G and H as its only "factors." There are many ways of defining this product and for completeness of our discussion we list some of these definitions below [61, 42, 7]. In each case, the vertex set of the product graph is defined to be the cartesian product of the vertex sets of the factors.

Definition 1. *Given undirected graphs G and H, the* **direct product** *of G and H denoted $G \times H$ is defined as follows:*

$$\begin{aligned} \mathcal{V}(G \times H) &= \mathcal{V}(G) \times \mathcal{V}(H) \\ \mathcal{E}(G \times H) &= \{\{(g_1, h_1), (g_2, h_2)\} : \{g_1, g_2\} \in \mathcal{E}(G) \text{ and } \{h_1, h_2\} \in \mathcal{E}(H)\}. \end{aligned}$$

Definition 2. *Given undirected graphs G and H, the* **cartesian product** *of G and H denoted $G \square H$ is defined as follows:*

$$\begin{aligned} \mathcal{V}(G \square H) &= \mathcal{V}(G) \times \mathcal{V}(H) \\ \mathcal{E}(G \square H) &= \{\{(g_1, h_1), (g_2, h_2)\} : \{g_1, g_2\} \in \mathcal{E}(G) \text{ and } h_1 = h_2, \text{ or} \\ &\qquad \{h_1, h_2\} \in \mathcal{E}(H) \text{ and } g_1 = g_2\}. \end{aligned}$$

Definition 3. *Given undirected graphs G and H, the* **Shannon product** *or* **strong product** *of G and H denoted $G \boxtimes H$ is defined as follows [42]:*

$$\begin{aligned} \mathcal{V}(G \boxtimes H) &= \mathcal{V}(G) \times \mathcal{V}(H) \\ \mathcal{E}(G \boxtimes H) &= \mathcal{E}(G \square H) \cup \mathcal{E}(G \times H). \end{aligned}$$

Definition 4. *Given undirected graphs G and H, the* **lexicographic product** *of G*

and H denoted $G \circ H$ is defined as follows [42]:

$$\begin{aligned}\mathcal{V}(G \circ H) &= \mathcal{V}(G) \times \mathcal{V}(H) \\ \mathcal{E}(G \circ H) &= \{\{(g_1, h_1), (g_2, h_2)\} : \{g_1, g_2\} \in \mathcal{E}(G) \text{ or, } g_1 = g_2 \text{ and } \{h_1, h_2\} \in \mathcal{E}(H).\end{aligned}$$

Definition 5. *Given undirected graphs G and H, the* **Sperner product** *of G and H denoted $G \odot H$ is defined as follows [7]:*

$$\begin{aligned}\mathcal{V}(G \odot H) &= \mathcal{V}(G) \times \mathcal{V}(H) \\ \mathcal{E}(G \odot H) &= \{\{(g_1, h_1), (g_2, h_2)\} : \{g_1, g_2\} \in \mathcal{E}(G) \text{ or } \{h_1, h_2\} \in \mathcal{E}(H).\end{aligned}$$

10.2 Motivation

Shannon defined the strong product in an information theoretic context. If $V(G)$ is interpreted to represent an alphabet, and $E(G)$ to represent the set of pairs of confoundable symbols, then an independent set in the k-th Shannon power of G represents a set of mutually nonconfoundable words. The cardinality of this set is no smaller than the cardinality of an independent set in G and could in fact be larger, thus allowing one to construct an alphabet of pairwise nonconfoundable symbols that is larger than the given $V(G)$. The extent to which a given alphabet can be powered up into a larger one is measured by the Shannon capacity of G, which is defined as follows.

Definition 6 (Shannon Capacity of a graph [104])**.** *Let $\alpha(G)$ denote the size of a largest independent set in G. The Shannon capacity of a graph G,*

$$\mathsf{shn}(G), \triangleq \lim_{k \to \infty} \frac{\log \alpha(G^{\boxtimes k})}{k}.$$

where $G^{\boxtimes k}$ is the k-th power of G under the strong product.

The problem of determining the Shannon capacity of even particular graphs (e.g., C_5 [78]) has turned out to be quite difficult and is open for even simple graphs (e.g.,

C_7). Alon and Lubetzky have asked whether this difficulty may be symptomatic of the underlying algorithmic complexity of the general problem:

Question 1. *Is it decidable whether the Shannon Capacity of a given graph exceeds a given value? [8]*

Similar questions have been asked about the computability of other invariants under other graph products:

Definition 7. *Let $\alpha(G)$ denote the size of a largest independent set in G. The independence ratio of a graph G,*

$$\mu(G), \triangleq \lim_{k\to\infty} \frac{\alpha(G^{\times k})}{|\mathcal{V}(G^{\times k})|}.$$

where $G^{\times k}$ is the k-th power of G under the direct product.

Question 2. *Is it decidable whether $\mu(G) > \beta$ for a given graph G and given value β? [9]*

In this chapter, we give an example of an undecidable problem for a type of graph power whose special case is the Shannon power. We also give an example of an undecidable problem about the cardinality of a special type of independent set under a different product. While this does not answer the questions raised above, it perhaps adds to the growing evidence for the difficulty of the problem.

10.3 Transpose walks in direct powers

Let us recall the definition of the direct product of two digraphs.

Definition 8 (Direct product [61]). *Given digraphs $G = (V(G), E(G))$ and $H = (V(H), E(H))$, the vertex set of the direct product $G \times H$ is $V(G \times H) = V(G) \times V(H)$ and the edge set $E(G \times H)$*

$$= \{((g,h),(g',h')) \,|\, g,g' \in V(G), h \neq h' \in V(H), (g,g') \in E(G), (h,h') \in E(H)\}.$$

The direct power generalizes the Shannon power: if G has a loop at every vertex, then its k-th direct power is also the k-th Shannon power. We first prove the following simple property before stating and proving our main result.

Definition 9. *For a digraph G, a* **walk** *in G is a nonempty sequence of vertices $v_1, \ldots, v_k$ in $V(G)$ such that $(v_i, v_{i+1}) \in E(G)$ for all $i \in \{1, \ldots, k-1\}$. The* **length** *of the walk is k.*

Proposition 1. *For digraphs X and Y, there exists a walk $P = p_1, p_2, \ldots, p_k$ in $X \times Y$ iff there exists a walk $Q = q_1, q_2, \ldots, q_k$ in X and a walk $R = r_1, r_2, \ldots, r_k$ in Y such that $\forall i \in \{1, \ldots, k\} : p_i = (q_i, r_i)$.*

Proof. (Only if): Since P is a walk in $X \times Y$, by definition of the direct product, $p_i = (x_i, y_i)$ for some $x_i \in V(X)$ and $y_i \in V(Y)$. Since P is a walk in $X \times Y$, $(p_i, p_{i+1}) \in E(X \times Y)$ for all $i \in \{1, \ldots, k-1\}$. By definition of the edge set of the direct product, it must be the case that $(x_i, x_{i+1}) \in E(X)$ and $(y_i, y_{i+1}) \in E(Y)$, for all $i \in \{1, \ldots, k-1\}$. Thus, $x_1, \ldots, x_k$ is a walk in X and $y_1, \ldots, y_k$ is a walk in Y.

(If): If $(q_i, q_{i+1}) \in E(X)$ and $(r_i, r_{i+1}) \in E(Y)$, then by the definition of the direct product, $((q_i, r_i), (q_{i+1}, r_{i+1})) \in E(X \times Y)$ for all $i \in \{1, \ldots, k-1\}$. Thus, $(q_1, r_1), \ldots, (q_k, r_k)$ is a walk in $X \times Y$. □

For the direct power, we show the following main result.

Theorem 1 (Transpose Walk problem). *Let A be an alphabet with $a \in A$ and let X, Y be digraphs with vertex set A. Let $G = X \times Y$, the direct product of X and Y and denote any $v \in V(G)$ as $(\pi_1(v), \pi_2(v))$. It is undecidable whether, for every $k \in \mathbb{Z}_{>0}$, there exists a walk $g_1, \ldots, g_k$ in $G^{\times k}$ such that*

1. *$g_{11} = (a, a)$, and*
2. *$\forall i, j \in \{1, \ldots, k\} : \pi_1(g_{ij}) = \pi_2(g_{ji})$*

where g_{ij} is the j-th component of the i-th vertex in the walk.

We prove the transposable walk problem (TWP) undecidable by showing that if it were decidable, then the Tiling problem (TP) would be decidable. The following is well-known:

Theorem 2 (Tiling problem [74]). *Let $\mathbb{T} = (T, H, V, t_0)$ be an instance of a tiling problem, where T is a finite nonempty set of tile types, $H, V \subseteq T \times T$, and $t_0 \in T$. The following problem is undecidable: Given an instance of a tiling problem, is there a function $f : \mathbb{N} \times \mathbb{N} \to T$ such that*

1. *$f(0,0) = t_0$, and*

2. *$\forall x, y \in \mathbb{N} : (f(x,y), f(x,y+1)) \in V$ and $(f(x,y), f(x+1,y)) \in H$?*

We now prove Theorem 1.

Proof of Theorem 1. We prove this by contradiction. Suppose that the TWP is decidable. Then there exists a Turing machine that decides every instance of the TWP. We show that this machine can then be used to decide every instance of the TP by converting an instance of the TP into an instance of the TWP while preserving the "yes/no" answer.

Given an instance of the Tiling problem $\mathbb{T} = (T, H, V, t_0)$, construct a graph $G = H \times V$, the direct product of the tiling relations. Let $A = T$, the tile type set, and $a = t_0$, the origin tile. Thus, the instance of the TWP constructed is the following: For every $k \in \mathbb{Z}_{>0}$, is there a walk $g_1, \ldots, g_k$ in $(H \times V)^{\times k}$ such that $g_{11} = (t_0, t_0)$ and $\forall i, j \in \{1, \ldots, k\} : \pi_1(g_{ij}) = \pi_2(g_{ji})$?

We must now show that a positive instance of the TP implies a positive instance of the TWP problem and a negative instance of the TP implies a negative instance of the TWP problem as per our construction.

Suppose a given instance of the TP is positive. Then there exists an infinite tiling of $\mathbb{N} \times \mathbb{N}$ given by $f : \mathbb{N} \times \mathbb{N} \rightarrow T$. In such a tiling, every $k \times k$ square of tiles that includes the origin obeys the tile abutment constraints specified by the relation V. That is, for all $i \in \{1, \ldots, k-1\}$, and all $j \in \{1, \ldots, k\}$, $(f(j,i), f(j,i+1)) \in E(V)$. Thus, for all $i \in \{1, \ldots, k-1\}$

$$\left(\left(f(1,i), f(2,i), \ldots, f(k,i) \right), \left(f(1,i+1), f(2,i+1), \ldots, f(k,i+1) \right) \right) \in E(V^{\times k}).$$

Thus, $(f(1,1), f(2,1), \ldots, f(k,1)), \ldots, (f(1,k), f(2,k), \ldots, f(k,k))$ is a walk in $V^{\times k}$. Similarly, for all $i \in \{1, \ldots, k\}$, and all $j \in \{1, \ldots, k-1\}$, $(f(j,i), f(j+1,i)) \in E(H)$: all the tiles obey the horizontal abutment rules specified by H. Thus, for all $j \in \{1, \ldots, k-1\}$

$$\left(\left(f(j,1), f(j,2), \ldots, f(j,k) \right), \left(f(j+1,1), f(j+1,2), \ldots, f(j+1,k) \right) \right) \in E(H^{\times k}).$$

Thus, $(f(1,1), f(1,2), \ldots, f(1,k)), \ldots, (f(k,1), f(k,2), \ldots, f(k,k))$ is a walk in $H^{\times k}$. Applying Lemma 1 to the walks

$$(f(1,1), f(2,1), \ldots, f(k,1)), \ldots, (f(1,k), f(2,k), \ldots, f(k,k))$$

in $V^{\times k}$ and

$$(f(1,1), f(1,2), \ldots, f(1,k)), \ldots, (f(k,1), f(k,2), \ldots, f(k,k))$$

in $H^{\times k}$, we can conclude that there exists a walk

$$\begin{array}{l} \left(\binom{f(1,1),}{f(1,1)}, \binom{f(2,1),}{f(1,2)}, \ldots, \binom{f(k,1),}{f(1,k)} \right), \\ \left(\binom{f(1,2),}{f(2,1)}, \binom{f(2,2),}{f(2,2)}, \ldots, \binom{f(k,2),}{f(2,k)} \right), \\ \quad \ldots, \\ \left(\binom{f(1,k),}{f(k,1)}, \binom{f(2,k),}{f(k,2)}, \ldots, \binom{f(k,k),}{f(k,k)} \right), \end{array} \text{ in } G^{\times k}.$$

We prove the second part by proving its contrapositive: a positive instance of TWP problem implies a positive instance of the TP. Suppose the given paths exist for all $k \in \mathbb{Z}_{>0}$. Then, for any $k \in \mathbb{Z}_{>0}$, the tiling of the $k \times k$ square of points can be obtained simply by tiling point (i, j) with the tile $x_i(j)$. □

10.4 Transpose Independence Ratio in Lexicographic direct product families

The lexicographic direct product is defined for directed graphs with loops.

Definition 10. *The* **lexicographic direct product** *of directed graphs G and H is the directed graph $G \otimes H$ with $\mathcal{V}(G \otimes H) = \mathcal{V}(G) \times \mathcal{V}(H)$ and*

$$\mathcal{E}(G \otimes H) = \{ \quad (\quad (g,h),(g',h') \quad) : g = g', \quad or \quad (g,g') \in \mathcal{E}(G) \text{ and } (h,h') \in \mathcal{E}(H) \quad \}.$$

Of course, this product is not commutative.

Proposition 2. *The lexicographic direct product is not associative.*

Proof. Let G, H, K be undirected graphs. Let (h, h') be an arc in H and (k, k') not be an arc in K, for some $h, h' \in \mathcal{V}(H)$ and $k, k' \in \mathcal{V}(K)$. Let $g \in \mathcal{V}(G)$. Observe that $e = (ghk, gh'k)$ is an arc in $G \otimes (H \otimes K)$. But it is not an arc in $(G \otimes H) \otimes K$ because although (gh, gh') is an arc in $G \otimes H$, (k, k') is not an arc in K. □

The lexicographic direct product of G and H has the following effect. Each vertex of G is replaced with a copy of the vertices in H. Vertices in each such copy are turned into cliques. Vertices in two such cliques are related to each other iff their corresponding "parent" vertices in G are related to each other. The specific pattern of relation between vertices in one copy with the vertices in another is determined by the arcs in H: H's vertices are cloned and an arc (a, b) in H is translated into an arc

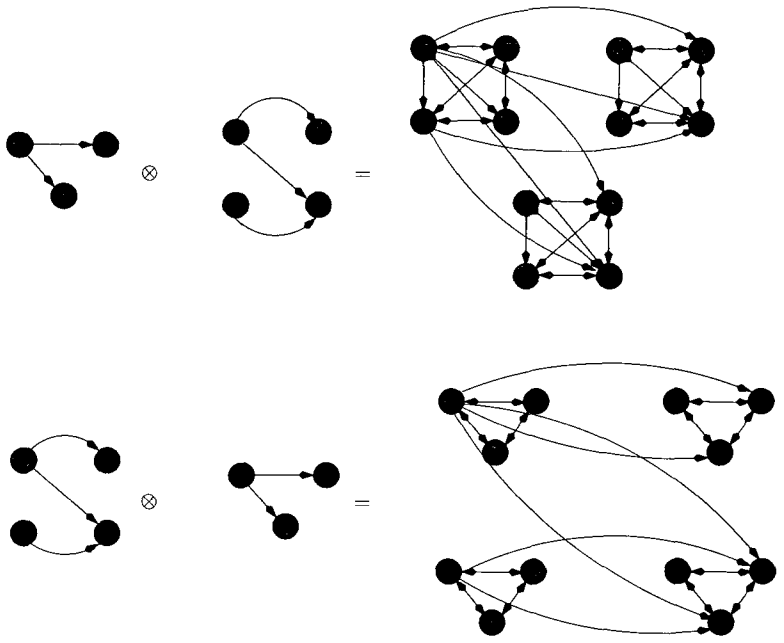


Figure 10-1: An illustration of the lexicographic direct product

between a in the first copy and b in the second. This is illustrated in Figure 10-1. As one might anticipate, these precise mechanics are vital to our next result.

Definition 11. *Let $\mathcal{A}$ be finite set with $a \in \mathcal{A}$ and let $X = \{0, \ldots, n-1\}$, $n > 0$. Let G be an directed graph with $\mathcal{V}(G) = \mathcal{A}^2 \times X^2$. An* **independent set** *in G is a set $S \subseteq \mathcal{V}(G)$ such that for all $u, v \in S$, $(u, v) \notin \mathcal{E}(G)$. An independent set S in G is called* **transposable** *iff $(a, a, 0, 0) \in S$ and for every $(p, q, x, y) \in S$, it is the case that $(q, p, y, x) \in S$.*

Definition 12. *Let $\alpha^T(G)$ denote the cardinality of the largest transposable independent set in G. The* **transpose independence ratio** *of an infinite family of directed graphs $\mathcal{G} = \{G_k : k \in \mathbb{Z}_{>0}\}$ is*

$$A^T(\mathcal{G}) = \lim_{k \to \infty} \frac{\alpha^T(G_k)}{|\mathcal{V}(G_k)|},$$

defined only for families where this limit exists and is finite.

The above definition is inspired by a similar invariant defined by Albertson et al. [5].

Theorem 3. *It is undecidable whether the transpose independence ratio of an infinite family of directed graphs $\mathcal{G}$ generated by the lexicographic direct product is at least a constant $c(\mathcal{G})$.*

Proof. Let P_k be the bidirectional path on k vertices and let U_k be the graph on k isolated vertices. Let H and V be horizontal and vertical abutment relations of a tiling system with n distinct types of tiles. Let $G = \overline{H \times V}$ and $c(\mathcal{G}) = 1/n^2$. Take $\mathcal{G} = \{U_k \otimes (P_k \otimes G) : k \in \mathbb{Z}_{>1}\}$. The family $\mathcal{G}$ then has a transpose independence ratio of at least $1/n^2$ iff the tiling system defined by H and V can tile the plane. □

10.5 Conclusion

These examples strengthen our belief in the conjecture that Shannon capacity and other invariants in other products may be uncomputable.

Chapter 11

Throughput-Delay Tradeoff in Small and Sparse Mobile Ad hoc Networks

This chapter takes the first step in characterizing the throughput-delay tradeoff for small and sparse MANETs which have many practical applications. We find that as the MANET becomes sparser, throughput decreases and delay increases, as expected. If relaying is disabled, then the throughput and delay depend on the size of the area of operation. While relaying does increase throughput, the single packet relaying strategy worsens the delay for small MANETs in the Grossglauser traffic model. Greedy relaying overcomes this worsening without trading throughput, but only for rapidly mixing mobility. Unlike in dense networks, local broadcasting does not provide any significant benefit. Packet repetition does decrease delay, but only at the expense of reduced throughput. Our results are useful in practical underwater MANETs which are typically small and sparse.

11.1 Motivation

The characterization of the throughput-delay tradeoff in wireless ad hoc networks has been the subject of study in a number of papers in recent years [109, 38, 55]. Most of the previous work in this area, except that of Spyropoulos et al. [109], has focused

on dense wireless networks with the tradeoff being studied as the number of network nodes n goes to infinity. A fundamental assumption in such work is that the wireless network under study is sufficiently dense, with the scaling behavior under increasing density being the subject of study. A motivating example justifying this assumption is the ad hoc sensor network where a dense deployment of sensor nodes is desirable. In contrast, the practical deployment scenario for many wireless ad hoc networks, particularly those involving mobile nodes, is such that while a dense deployment is desirable, it is rarely feasible. Consider a MANET of autonomous underwater vehicles (AUVs) deployed for bathymetry or underwater surveillance. Even for such basic underwater missions, the oceanic region involved is far too vast to be amenable to sensing and measurement by a dense MANET. As a result, practical AUV MANETs tend to be small and sparse for which extant capacity results studying scaling behavior as a function of increasing density provide little insight into the tradeoffs involved in such networks. A fundamental differentiating characteristic of a sparse MANET is the high probability with which a mobile node may be outside the transmission range of any other node. Whereas the interference among concurrent transmissions plays a deciding role in the throughput-delay tradeoff in dense networks, such interference is rare in sparse networks. In what other respects might sparse MANETs be different from dense ones? This is the motivating question for our work and this chapter takes the first step towards answering it.

11.2 Model

We model the spatial region in which the mobile nodes of a sparse MANET move as a discrete undirected graph with loops. We experiment with two graphs: the complete graph on m^2 vertices and the $m \times m$ two-dimensional torus. We consider $n \geq 2$ mobile nodes performing a random walk on the underlying graph at each discrete

time step. Each mobile node i produces data packets destined for exactly one other node denoted $\mathsf{dest}(i)$. This source-destination mapping is fixed and is chosen by a random derangement of $\{1, \ldots, n\}$. At any time step, a node has exactly one packet available for transmission. After a packet has been transmitted, a new packet is available for transmission immediately as in the traffic model of Grossglauser et al. [55]. If a set M_v of (more than one) mobile nodes meet at any vertex v, then data transmission occurs according to the following rules:

1. Every mobile node $i \in M_v$ such that $\mathsf{dest}(i) \in M_v$, transmits a single packet to $\mathsf{dest}(i)$. We call this a *direct delivery.*

2. Every node $i \in M_v$ that could not perform a direct delivery chooses at random a $j \in M_v$ for which it carries one or more packets delegated to it by j's source. It then delivers at most p such packets to j, where the particular packets transmitted, if more than p are available, are also chosen at random. We call this a *relayed delivery.*
3. Every node $i \in M_v$ that could not perform a direct or relayed delivery transmits exactly one packet to another randomly chosen node $j \in M_v$ requesting j to deliver the packet to $\mathsf{dest}(i)$. We refer to this as *packet delegation.*

When $|M_v| > 1$, each node either makes a direct or relayed delivery or delegates a packet. Each node possesses infinite space for storing delegated packets that it has accepted. Transmissions occur in a round-robin manner and are coordinated through some TDMA scheme at each vertex v. Transmission of a single packet takes a constant amount of time and the total time spent in communication at each vertex is negligible in comparison to the inter-vertex travel time. Since the network is sparse, transmissions occur concurrently at all vertices v without mutual interference.

The above rules are similar to those used in Grossglauser et al. [55]. In addition, we study the following variants. When *delegation* is disabled, a node is only capable

of direct delivery via method (1) above. When delegation is enabled and $p = \infty$ in method (2), we call this *greedy relaying.* If *local broadcasting* is enabled, then a packet transmitted by a node i via method (3) is broadcasted to all nodes $j \in M_v$, i.e., i delegates the packet to all other nodes present at v through a single transmission [38]. If *packet repetition* is enabled with parameter r, then every packet produced by a node i is delegated $1 + r$ times by i or until it is delivered directly, whichever occurs earlier.

11.3 Simulation results

We simulated our model of a sparse MANET on a complete graph and a torus and measured the average throughput and delay. We adopt a natural definition for sparsity; it is the difference in the orders of magnitude of the number of mobile nodes (n) and the number of vertices in the underlying graph (m^2).

Figure 11-1 and Figure 11-2 show the throughput-delay tradeoff on a complete graph and the torus for a fixed $m = 100$ with n varied between 2 and 500. We can make the following observations, the most striking of which is that while packet delegation increases throughput, it worsens delay, for high sparsity. This is true for both the complete graph as well as the torus. Moreover, whereas greedy relaying mitigates this rise in delay for the complete graph, it is ineffective on the torus. Except for the delay when delegation is disabled, the delay for the torus is higher than that for the complete graph when the sparsity is high.

While throughput improves, the average delay when relaying is enabled is worse than when it is disabled. This is mainly an artifact of using the Grossglauser traffic model. In this model, a node has a new packet available for transmission immediately after it has transmitted the previous one. When relaying is disabled, only the source can transmit packets to its destination. Thus, a new packet becomes available

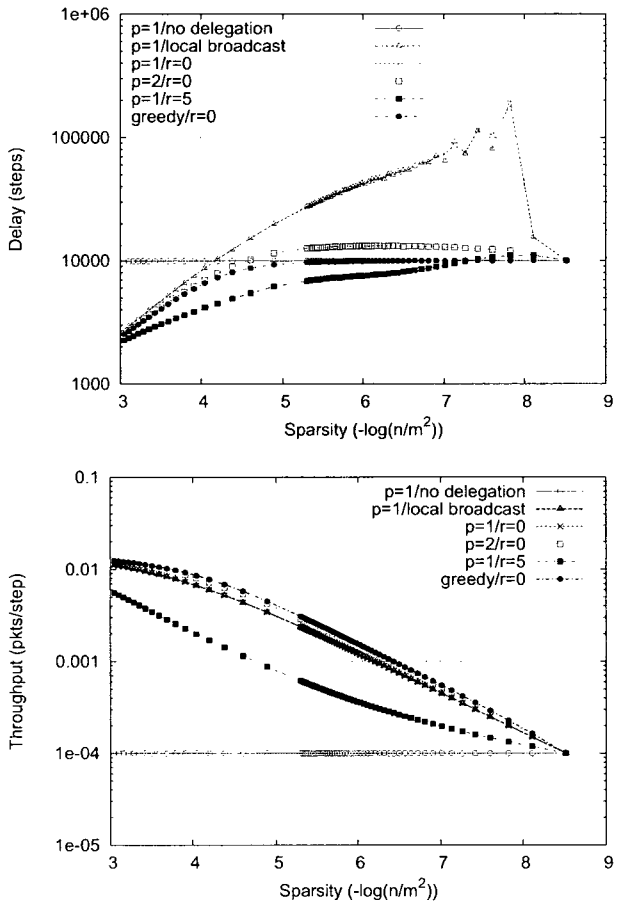


Figure 11-1: MANET on a complete graph when n is varied with $m = 100$.

only and immediately after the source delivers its previous packet. When relaying is enabled, a source can generate a new packet as soon as it has delegated the previous one to a relay. Thus, a larger number of packets may wait in the queue of various relays. Moreover, if $p = 1$, then a relay carrying packets for $\mathsf{dest}(j)$ can only deliver one packet to $\mathsf{dest}(j)$ per meeting, as in the Grossglauser model. Furthermore, the chance that a packet for $\mathsf{dest}(j)$ will be chosen via method (2) is lowered if the relay has packets for several destinations which it meets at the same time. Thus, a relay carrying packets for $\mathsf{dest}(j)$ may have to meet $\mathsf{dest}(j)$ multiple times to deliver all the waiting packets that it has queued up for it. This type of queuing is impossible when relaying is disabled: a packet's waiting time starts only when it is generated and it is generated only when the previous packet has been delivered.

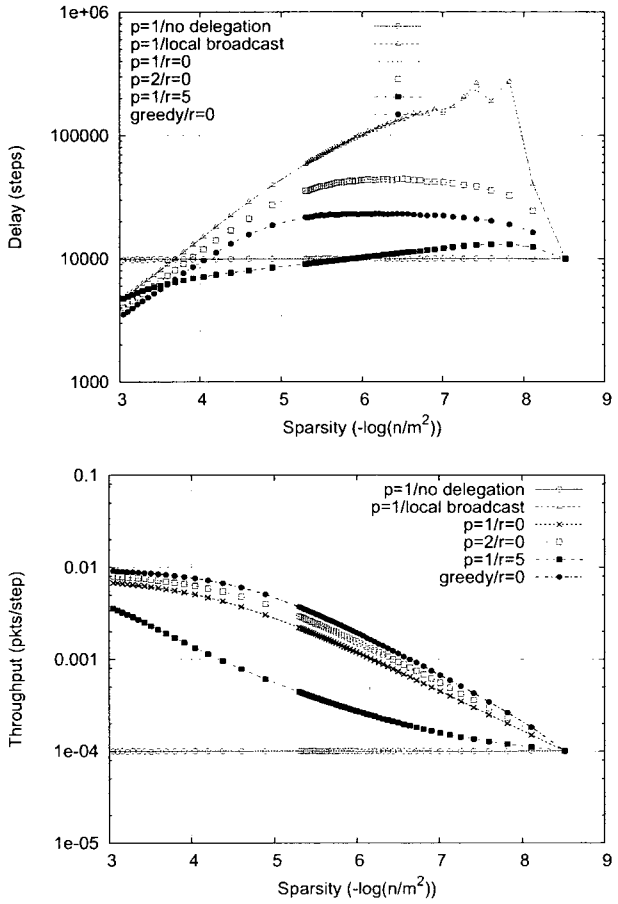


Figure 11-2: MANET on a two-dimensional torus when n is varied with $m = 100$.

It is no surprise that greedy relaying is effective in mitigating the rise in delay discussed above. With greedy relaying, a relay can deliver all the packets in its queue to $\mathsf{dest}(j)$ in a single meeting, provided that $\mathsf{dest}(j)$ is randomly chosen for delivery via method (2). However, whereas greedy relaying lowers the delay to that achieved without relaying on the complete graph, it is not as effective on the torus. The delay for the torus is also higher than that for the complete graph. The mixing and meeting time of the torus is $\Theta(m^2)$ and $\Theta(m^2 \log m^2)$ respectively while that of the complete graph is $\Theta(1)$ and $\Theta(m^2)$ respectively [6]. Because of this, the time to meet a node's destination is also higher on the torus. Also, it is more likely on a torus that a node will meet the same relay multiple times in quick succession, each time delegating a new packet to it. Because of this, packets are not spread out uniformly across many

different relays. Hence, the fate of many packets may end up relying on the meeting time of a few relay nodes. These factors conspire together to increase the average number of packets carried by a relay on the torus for a particular destination, as compared to the number carried by a relay on the complete graph. Thus, a larger number of packets accrue the inherently higher delay of the torus due to its meeting and mixing time, and this leads to a higher delay despite greedy relaying. Note that when the delay is higher, the corresponding throughput observed on the torus is also slightly higher than the throughput on the complete graph.

Packet repetition is highly effective in decreasing the delay because it increases the likelihood that packets are spread uniformly across a larger number of different relays. This decrease comes at the price of lowered throughput. Local broadcast has no discernible effect on delay or throughput because $|M_v| > 2$ necessary for local broadcast to occur is a rare event in sparse MANETs.

11.4 Conclusions and future work

In this work, we reported our results on the throughput-delay tradeoff for sparse MANETs which show that the scaling behavior of small and sparse networks is different from dense networks studied in the past. What is the effect of a traffic model that is more natural than the Grossglauser model used here? What kind of motion, graph, and traffic model captures practical AUV missions? We intend to investigate such questions about sparse networks more deeply in our future work.

Part III

Prefetching in storage networks

Chapter 12

Prefetch cache sizing: Optimal is impossible but near-optimal is feasible

This chapter focuses on the sizing requirements of the prefetch cache partition of a disk array cache, and derives the minimum size (*optimal size*) required to get the maximum hit rate for a workload. We then formulates and answers the following question: *Can the prefetch cache size in a storage system be matched optimally to a workload?* It is shown that while it is impossible for a storage prefetch cache to maintain the optimal size and get the maximum prefetch hit rate for the workload, it is possible to maintain a size that is a factor greater than the optimal and achieve a hit rate close to the maximum hit rate. For example, keeping the prefetch cache size twice greater than the optimal achieves more than 85% of the maximum hit rate; and if the size is increased to thrice greater than the optimal, it is possible to achieve more than 95% of the maximum hit rate. A key result shows that increasing the cache size beyond the optimal for a prefetching technique only provides an exponentially small gain in hit rate. We develop a new prefetch cache sizing technique that dynamically adjusts the size of the prefetch cache based on the workload's changing sequentiality. This sizing scheme is then incorporated into standard sequential prefetching techniques. Our experimental results show that the sizing technique keeps the prefetch cache size

a factor greater than the optimal while achieving more than 98% of the maximum hit rate.

12.1 Introduction

Disk arrays consist of several disks, an array controller, and one or more array caches. In addition to increasing the capacity and availability of storage systems, disk arrays improve the speed of storage access by distributing the load across disks. However, it still takes milliseconds to access data from the disks, so the speed differential between host computers and storage is still substantial. The speed of disk array data access can be improved significantly if data are already loaded in the array caches when I/O requests for the data arrive. Unlike disk caches which are small, array caches are large and are often larger than file system caches. Unlike disk controllers that are only capable of doing basic tasks, disk array controllers are capable of performing complex tasks to speed data access. The size of the array cache along with the power of the array controller gives the disk array the hardware capability of running complex caching/prefetching techniques capable of speeding the storage device.

Prefetching techniques speed up storage access by loading data from the disks into the cache before I/O requests for the data arrive. In order to determine what data have to be prefetched into the cache, a prefetching technique has to predict the future data needs of applications accessing storage. Several applications may be accessing storage at a time. Each application may have several files open for reading at a time, or may open a file more than once. Each open command generates a **stream** of I/O requests to a single file over a period of time. The operating system submits the outstanding requests from the various streams to the storage system. An I/O request submitted to the storage system consists of the request's address, whether the request is of type read/write and the number of blocks to be accessed.

The storage system is not provided any information on applications, streams or files, so it does not know which stream relates to an I/O request. A storage controller does not know when a new stream starts and when an existing stream closes. Therefore, a storage prefetching technique has to infer the future data needs of applications using only I/O request addresses.

The "narrow" I/O interface limits the prediction capabilities of storage prefetching techniques [76]. Luckily, file data are often read sequentially, and operating systems try to store a file's data contiguously on storage disks [41, 100]. Therefore, there is a non-zero probability that some of the streams accessing storage are sequential or partly sequential. Consequently, sequential prefetching is the most common prefetching technique implemented in storage systems [46, 47, 54, 75, 114]. A sequential prefetching technique generates prefetch requests for data that are stored contiguously to I/O request data. A central goal of a prefetching scheme is to keep prefetched data in the cache until I/O requests for the data arrive.

In addition to prefetched data, an array cache is used for temporarily storing I/O write request data and I/O read request data. While an array cache is large compared to a disk cache, space is still a premium since the size of an array cache is just 0.1% to 1% of the storage capacity [41, 113]. One of the issues that need to be addressed is the space requirements of a prefetch cache. Prefetch data are brought in on the assumption that I/O requests will eventually arrive. If the prefetch cache is too small, prefetched data may get ejected from the cache before I/O requests for the data arrive. Ejecting useful prefetched data results in a double whammy performance drop, since the data have to be reloaded from the disks when I/O requests corresponding to the ejected prefetched data arrive. The performance of a storage system would improve if the cache size is sufficiently large so that useful prefetched data remain in the cache long enough to result in cache hits. However, the prediction capabilities of a storage

prefetching technique are limited, so it is possible that some of the prefetched data are wrongly prefetched and never receive hits. This is especially true when the I/O workload contains several random streams and a "blind" prefetching technique like Prefetch-Always (that prefetches data contiguous to every I/O request) is used. If the prefetch cache is too large, then a lot of space is wasted storing wrongly prefetched data. Thus, the size of a prefetch cache plays a key role in the performance of a storage system.

Prefetching has been around since the dawn of computer systems, so there are numerous papers in this area [11] However, the majority of papers focus on the specifics of prefetching, namely, what to prefetch, when to prefetch, and how much to prefetch. The sizing of the prefetch cache has been largely ignored. This is the first work that systematically addresses storage prefetch cache sizing and analyzes the impact of sizing on prefetch hit rate for various prefetching techniques. The contributions of this work are the results on prefetch cache sizing and a new technique that dynamically sets prefetch cache size. A key result proved here shows that as the prefetch cache size increases beyond a certain optimum, the increase in hit rate decreases exponentially. Some of our results, like the impossibility of attaining maximum hit rate using optimal size, are negative. However, it is important to understand limits, so that one does not waste time trying to achieve the impossible. We also address the question: Is it possible for a storage controller to make informed/intelligent decisions on prefetch cache size, given that storage systems have no information on files or streams? Fortunately, we answer this question in the affirmative and present a new technique that sets the prefetch cache size to within a constant factor of the optimal cache size while achieving a prefetch hit rate that is very close to the maximum achievable hit rate for the submitted workload. This work does not develop a new prefetching technique - we develop a prefetch cache sizing scheme that can be incorporated into any

storage prefetching technique. While our theoretical derivations make assumptions like uniform interleaving of workload streams and geometric distribution of runs, our experimental validation of the results test out workloads that do not follow these assumptions.

The rest of this chapter is organized as follows: Section 12.2 describes the I/O workload. Section 12.3 proves the impossibility theorem. Section 12.4 computes the maximum prefetch hit rates of various prefetching schemes. Section 12.5 shows the possibility of suboptimality. Section 12.6 presents our dynamic cache sizing technique, and Section 12.7 validates the efficiency of our technique.

12.2 I/O workload

A prefetching technique generates **prefetch requests** for data in anticipation that **on-demand requests** for the prefetched data will arrive in the near future. A cache contains data loaded into the cache as a result of on-demand requests and prefetch requests. To clarify this distinction, we partition the **read cache** into the **on-demand cache** and the **prefetch cache**. The on-demand cache stores data that are loaded into the cache as a result of on-demand read requests, while the prefetch cache stores data that are loaded into the cache as a result of prefetch requests. When an on-demand request arrives for prefetched data, we assume that the hit prefetched data are removed from the prefetch cache. The hit data could either be moved into the on-demand cache or removed altogether from the read cache.

Sequential prefetching techniques generate prefetch requests for data that are stored contiguously to on-demand request data. This type of prefetching is common in file and storage systems since a large number of applications read files sequentially and many file systems try to store a file's data contiguously [41, 100]. A **stream** refers to I/O requests for a particular file's data. A new stream starts when a file is opened

for reading and the stream stops when the file is closed. During the life time of the stream, suppose $n > 0$ requests are generated. The stream consists of a sequence of I/O requests and is written as $< io_1, io_2, \ldots, io_i, io_j, \ldots io_n >$. At any point in time, a stream may have *outstanding* requests (*i.e.,* requests submitted to storage waiting to be served). Suppose request io_c from the request is currently outstanding. Then, requests $io_1, io_2, \cdots, io_{c-2}, io_{c-1}$ have been served, and requests io_{c+1}, io_{c+2} .. io_n are yet to be generated.

Definition 9. *Let io_i and io_j be two requests from the same stream. Request io_j is said to be* **sequential** *to request io_i if and only if $j = i + 1$, and io_j's data are stored contiguously following io_i's data on the storage device. Request io_j is said to be* **random** *to request io_i if and only if $j = i + 1$, and io_j's data are* not *stored contiguously following io_i's data on a storage device.*

By definition, the first request of an I/O stream is neither sequential nor random. Without loss of generality, we set the first request in the stream to be a random request. All other requests are either sequential or random. From the viewpoint of sequentiality of an I/O stream, if request io_j is sequential to request io_i, then io_j is written as s, else io_j is written as r. Thus, the first request in a stream is always r, and the following requests are s or r depending on whether they are sequential or random to the previous request in the sequence. An I/O stream is then viewed as a sequence of requests of the form $< r, [r|s]^* >$. A **completely random** stream is of the form $< r, r, r, r, \ldots.. >$ while a **completely sequential** stream is of the form $< r, s, s, s, s, \ldots.. >$. A **partly sequential** stream is of the form like $< r, r, s, s, s, r, r, r, s, r, \cdots >$. Every subsequence of contiguous sequential requests in a partly sequential stream is referred to as a **sequential run**. A **sequential stream** refers to a stream that is completely or partly sequential.

Interleaved streams: An I/O workload consists of interleaved requests from various streams. Suppose an I/O workload consisting of requests from 5 streams. The

operating system is stream-aware, so an OS prefetching technique sees the I/O workload as an interleaving of requests from the 5 streams of the form: $< st_1, st_3, st_2, st_5, st_1, \ldots, st_4, st_4, st_1, \ldots >$, where st_i represents an I/O request from stream i. However, a storage system is not given any stream information, so a storage prefetching technique sees the I/O workload as $< io_1, io_2, io_3, io_4, \cdots >$. Since requests from various streams are interleaved, two (possibly sequential) requests from the same stream are typically separated by several requests from other streams. Suppose all the 5 streams are completely sequential. The OS sees the workload as $< \cdots, s_1, s_2, s_1, s_5, s_4, \cdots >$ where s_i represents a sequential request from stream i. Consequently, if the request-stream correspondence is known, then the sequentiality in the workload can be identified even if the requests from the streams are interleaved. However, a storage system prefetching technique just sees $< r, r, r, r, \cdots >$, a single stream of random requests. Thus, even if each file's data are stored contiguously on disks and applications are accessing these files sequentially, the I/O workload appears random to a storage system. All is not lost,however, since it is possible for a storage prefetching technique to infer stream information. For example, a list of past I/O request addresses can be kept and an incoming request's address can be compared to this list. If the incoming request address is contiguous to a past I/O request's address, then the new request is identified to belong to the same stream as this past request.

Thus, the operating system has information on request-stream correspondence while the storage system does not. At best, a storage system can infer this request-stream correspondence. In the next section, we prove that lack of precise information on the request-stream correspondence leads to storage prefetch cache sizes having to be maintained larger than the minimum size in order to get maximum hit rate for the workload.

12.3 Optimal Size

Suppose an I/O workload consists of M randomly interleaved streams of which $\mathsf{M_s}$ are sequential (completely or partly) and $\mathsf{M_r}$ are random. A prefetching technique that ensures that each of s requests in the $\mathsf{M_s}$ streams gets a hit achieves the **maximum prefetch hit rate** for the **workload**. The size of a prefetch cache is the number of cache lines in the cache. Assume that each cache line is capable of holding at least one prefetch request.

Definition 10. *The* **optimal** *prefetch cache size is the minimum size required to get the maximum hit rate for a workload.*

In order to get the highest possible hit rate for a workload, a sequential prefetching technique has to prefetch s requests pertaining to all sequential streams and keep the data in the prefetch cache until their on-demand requests arrive. The next theorem shows the relationship between a workload's maximum prefetch hit rate and the optimal prefetch cache size.

Theorem 9. *The optimal prefetch cache size is M_s, the number of sequential streams in the workload.*

Proof. In order to prove this theorem, we need to show that (a) if the prefetch cache size is equal to $\mathsf{M_s}$, then a prefetching technique can ensure that every sequential request s in the workload gets a hit; and (b) if the prefetch cache size is less than $\mathsf{M_s}$, then no prefetching technique can guarantee that every sequential request gets a hit.

We first prove (a). Suppose the prefetch cache size is equal to $\mathsf{M_s}$. Assume that the prefetching technique is aware of streams and their sequentiality. While the prefetching technique has no knowledge of the future workload, it knows which stream each request corresponds to. This is a reasonable assumption since program based prefetching techniques and file system prefetching techniques use stream information.

The prefetching technique is **stream-aware**. A new cache line is reserved when a new sequential stream is opened. A cache line is removed from the prefetch cache

when a sequential stream is closed. Prefetch requests are generated on a stream-basis, and are loaded into the cache line reserved for the corresponding stream. (For pedagogical reasons, assume that per-stream prefetch data fits into a single cache line.) A stream-aware prefetching technique that prefetches every sequential request and ensures that the data are in the prefetch cache before the on-demand request arrives will get the maximum hit rate for the workload.

We now prove (b). Suppose the prefetch cache size is less than $\mathsf{M_s}$. Then, there is at least one sequential stream i, whose request has not been prefetched into the cache. Since the workload consists of M randomly interleaved streams, there is non-zero probability that the next on-demand request is a sequential request from stream i. This request will miss in the prefetch cache. □

A partly sequential stream is composed of random and sequential requests. If the future workload is completely known, a cache line need not be reserved for sequential streams that are currently submitting random run requests.

Corollary 1. *A stream-aware prefetching technique that has complete knowledge of the future workload can achieve the maximum workload hit rate with a prefetch cache size less than $\mathsf{M_s}$, by reserving a cache line only for the streams that are currently submitting sequential requests.*

Corollary 2. *A stream-aware prefetching technique that is not aware of stream sequentiality can achieve the maximum workload hit rate with a prefetch cache of size M.*

Therefore, a file system prefetching technique that prefetches without identifying stream sequentiality can achieve the maximum prefetch hit rate with a prefetch cache of size M. The prefetching techniques that achieve maximum hit rates using minimal size caches have **stream-aware cache replacement** schemes. A cache line is set aside for every stream in the workload, and prefetched data must be loaded into the

cache line reserved for the corresponding stream. Prefetched data that receive a hit are moved out of the prefetch cache. When a new stream opens, a new cache line has to be added to the prefetch cache, and when a stream closes, its cache line is removed from the cache.

Cache eviction policy that infers stream information: Storage systems have no knowledge of streams, and consequently do not know of the correspondence between requests and streams. The cache replacement policy for storage prefetch caches are typically based on the Least-Recently-Used (LRU) or First-In-First-Out (FIFO) schemes [27, 46, 47, 64, 94]. In fact, the on-demand cache and the prefetch cache are often treated as one unit and a replacement policy is used on the entire read cache [35, 48, 95, 106, 107]. Therefore, storage prefetching techniques are not stream aware and are unable to reserve cache lines on a stream basis. Consequently, a storage cache replacement scheme cannot guarantee that a new prefetch request is loaded into the same cache line where the prior prefetch request from this stream was loaded. However, it is possible for storage systems to infer streams in the I/O workload using cache replacement schemes that are variations of the Least-Recently-Used (LRU) technique as explained below. When a new prefetch request has to be inserted and the cache is full, the prefetch request at the LRU head gets evicted from the cache, and the free cache line is moved to the Most-Recently-Used (MRU) end of the cache. The new prefetch request is loaded into this empty cache line at the MRU head. When a prefetch request gets a hit, the request is moved out of the prefetch cache and the cache line is moved to the MRU head of the cache. This cache line may now be empty or partly empty (if the prefetch scheme prefetched data relating to more than 1 on-demand request). Depending on the prefetching technique, this empty or partly empty cache line could be used to store more prefetch data contiguous to the hit request. The replacement scheme is like the First-In-First-Out (FIFO) scheme,

except that every time there is a hit, the hit data are moved out of the cache and the cache line is moved to the MRU head.

At best, the above scheme can infer sequential stream information by prefetching data contiguous to a previous hit. However, without information on request-stream correspondence, it is impossible to know when streams start and stop, and when a sequential run in a partly sequential stream ends or when a new sequential run in a partly sequential stream starts. The storage prefetch cache replacement policies are non stream aware and most of the replacement policies are variations of the LRU scheme [27, 46, 47, 64, 94]. The following theorem proves the impossibility of storage prefetch caches maintaining optimal size and getting the maximum hit rate for the workload.

Theorem 10. *For a cache employing a LRU/FIFO based eviction policy, no prefetching technique using an optimal sized prefetch cache can achieve the maximum hit rate for a workload of arbitrarily interleaved streams, where some of the streams are partly sequential.*

Proof. In order to prove this theorem, we need to show that there is non-zero probability that a sequential request misses in the cache. Let the prefetch cache size be set to $\mathsf{M_s}$, the optimal size. Suppose a sequential request s_i from stream i is prefetched into the MRU end of the cache. There are $\mathsf{M_s}$- 1 cache lines separating the MRU head and the LRU head of the cache. Since the streams are interleaved, there is non-zero probability that the next on-demand request submitted for a sequential stream is from stream j, where $i \neq j$. Two cases can arise:

Case 1: The request from sequential stream j is a sequential request s_j (*i.e.*, the request is part of a sequential run). If s_j misses in the prefetch cache then the theorem is proved. If s_j hits in the prefetch cache, then the request is moved out of the corresponding cache line, and the newly empty or partly empty cache line is moved to the MRU head. If the cache line is not empty, then data contiguous to s_j is already

loaded in the prefetch cache. If the cache line is empty, a prefetching technique may generate a request for data contiguous to s_j. Since there is non-zero probability that s_j is not the end of a sequential run, a prefetching technique that does not generate such a prefetch request could result in a cache miss, thereby proving the theorem. This prefetched data for stream j would be loaded into the newly empty cache line at the MRU head. Thus, whenever there is a prefetch cache hit, the next prefetch request can be loaded into the prefetch cache at the MRU head without evicting the request from the LRU head. If the hit cache line is closer to the LRU head than the cache line storing s_i, the hit results in the cache line storing s_i moving one line closer to the LRU head of the cache.

Case 2: The request from sequential stream j is a random request r_j (*i.e.,* the request is part of a random run). There is non-zero probability that the next on-demand request from this stream could be for data contiguous to r_j (*i.e.,* a sequential run starts). A prefetching technique may generate a prefetch request for data contiguous to r_j. This prefetch request is generated as a result of a miss in the prefetch cache, so there are no newly emptied cache lines. As a result, the new prefetched data would result in the eviction of data from the LRU head of the cache. Thus, each new prefetch generated without a prefetch cache hit moves the cache line storing s_i one line closer to the LRU head of the cache.

Since the streams are randomly interleaved, there is a non-zero probability that the on-demand request s_i arrives only after the prefetch cache line corresponding to s_i has reached the LRU head and has been ejected. Thus, it is possible that prefetch request s_i gets evicted from the cache before the on-demand request s_i arrives. □

Storage systems are not stream-aware, and therefore use non stream-aware cache replacement techniques such as LRU. Theorem 10 proves that a storage system prefetching technique cannot achieve the maximum workload hit rate with a opti-

mal sized cache. In Section 12.5, we answer the question: what is the hit rate that can be achieved by an optimal sized storage system prefetch cache. Before addressing the above question, we outline the standard prefetching techniques implemented in storage system caches since the type of prefetching technique is relevant to the answer.

12.4 Sequential Prefetching Schemes

Sequential prefetching techniques are classified into the following: Prefetch Always (PA), Prefetch On a Miss (PoM) and Prefetch On a Hit (PoH). For clarity of explanation, in this section, it is assumed that data contiguous to only 1 request is prefetched. (For a pseudocode of these prefetching schemes, please see appendix.)

PA: Each arriving I/O request, regardless of whether the request hits or misses in the read cache, results in prefetch of data contiguous to the request's data. The advantage of PA is that every request from sequentially accessed files are prefetched, resulting in a high hit rate when the I/O workload contains a large number of sequential streams. However, when the workload consists of a large number of random streams, PA would prefetch large amounts of data that never receive prefetch hits.

Scheme 1 PREFETCH ALWAYS (PA)

1: **if** request io is an on-demand and prefetch cache miss **then**
2: Generate a single piggybacked request for io and $io + 1$
3: **else**
4: Serve io; if io in prefetch cache, then evict it from the prefetch cache; prefetch $io + 1$
5: **end if**

PoM: Each arriving I/O request that misses in the read cache, results in prefetch of data contiguous to the missed request's data. The advantage of PoM is that the prefetch request can be *piggybacked* onto the on-demand request. Therefore, instead of generating a separate on-demand request and a prefetch request, a single request for both the on-demand data and the prefetch data can be submitted to the disks. If the workload consists of a large number of random requests, then PoM results in a large number of prefetches that never receive hits. Another drawback is that if the workload consists of a large number of sequential streams, PoM misses prefetching half the requests in the sequential streams.

Scheme 2 PREFETCH ON A MISS (POM)

```
1: if request io is a on-demand and prefetch cache miss then
2:     Generate a single piggybacked request for io and io + 1
3: else
4:     Serve io; if io in prefetch cache, then evict it from the prefetch cache
5: end if
```

PoH: Each arriving request that hits in the prefetch cache results in prefetch of data contiguous to the hit request's data. Thus, PoH assumes that the prefetch cache contains sequential stream data. Every request that misses in the prefetch cache activates the *sequential access pattern detection module* which searches past request addresses to check if this missed request's data are contiguous to a prior request's data. If a past request address is found to be contiguous to this missed request's address, then PoH assumes that the missed request's data are from a sequential stream. Therefore, PoH prefetches data contiguous to the missed request's data by generating a piggybacked request for the on-demand data and prefetch data. PoH is the only technique that initiates a prefetch after identifying that a request is part of

a sequential file access. The prefetch data are stored in the prefetch cache while the on-demand data are transmitted upward and a copy of the data may also be written into the on-demand cache.

Scheme 3 PREFETCH ON A HIT (PoH)

1: **if** request io is a prefetch cache miss **then**
2: **if** request $io - 1$ is a on-demand cache hit **then**
3: Generate a single piggybacked request for io and $io + 1$; store io in on-demand cache and $io + 1$ in prefetch cache;
4: **else**
5: Serve io and save it in the on-demand cache
6: **end if**
7: **else**
8: Serve io, evict it from the prefetch cache, save io in on-demand cache, prefetch $io + 1$
9: **end if**

12.4.1 Prefetch Hit Rate

For a given prefetch cache size and given workload, the three types of prefetching techniques may give different hit rates. Below, we analyze the maximum hit rates achievable for the prefetching techniques for a given workload. The analysis assumes that 1) the prefetch cache size is sufficiently large, so that a prefetch request from a sequential stream does not get evicted from the prefetch cache before its on-demand request arrives; and 2) a sequential prefetch request arrives at the prefetch cache before its on-demand request arrives. Thus, the maximum hit rate computations assume that every sequential request that is prefetched will get a hit. Without these assump-

tions holding true, no prefetching technique can achieve the theoretical maximum hit rate for a given workload.

A single stream: The I/O workload contains interleaved requests from various streams. We first define parameters for a single stream in the I/O workload. In other words, the definitions and computations below are valid for a single stream in the interleaved I/O workload. Define the following parameters:

N(s): number of sequential requests in an I/O stream containing n requests;

N(r): number of random requests in an I/O stream containing n requests;

N(p): number of prefetched requests with respect to an I/O stream;

N(h): number of prefetched requests that result in a hit;

Definition 11. *The* **sequentiality**, S, *of the stream is given by* $\mathsf{S} = \mathsf{N(s)}/n$, *while the* **randomness**, R, *of the stream is given by* $\mathsf{R} = \mathsf{N(r)}/n = 1 - \mathsf{S}$.

Definition 12. *The prefetch* **hit rate**, H, *is given by* $\mathsf{H} = \mathsf{N(h)}/n$.

Only prefetched data pertaining to sequential requests can get a hit, so the **maximum prefetch hit rate** that can be achieved for an I/O stream is S and $\mathsf{H} \leq \mathsf{S}$.

Definition 13. *For an I/O stream, define a* **sequential run** *to be the sub-sequence of one or more consecutive sequential requests between two random requests. Thus, a sequential run includes all the sequential requests in the following sub-sequence* $< \ldots, r, s, s\ , s, \ldots, s, r, \ldots >$.

Definition 14. *A* **run** *is defined to be all the requests in a sequential run followed by all the requests before the start of the next sequential run. Thus, in the following sub-sequence* $< \ldots, r, s, s, \ldots, s, r, \ldots, s, \ldots >$, *a run includes all the sequential requests in the first sequential run and all the random requests until the next sequential run.*

Definition 15. *The requests between the end of a sequential run and the beginning of the consecutive sequential run is defined to be a* **random run**. *Thus, each run*

consists of a sequential run and a random run, and the number of runs in a stream is equal to the number of sequential (random) runs.

As per the above definitions, the minimum length of a sequential run is 1 and the minimum length of a random run is 1. Note that a stream's requests are interleaved with other stream's requests. So, sequential run requests from a stream may not arrive contiguously at the storage system. Suppose a stream i has a sequential run of length 3 that is of the following form: $< r, s_1, s_2, s_3, r >$. Since requests from various streams arrive at the storage system in an interleaved fashion, storage may see the following: $< x, r, x, s_1, x, x, x, s_2, x, s_3, x, x, r >$, where x represents requests from other streams. Regardless of how the requests arrive at the storage system, the sequential run for stream i starts when s_1 is submitted and ends when s_3 is submitted, so the length of the run is 3.

Definition 16. *A* **stream** *with* $0 < \mathsf{S} \leq 1$ *is* **sequential** *while a stream with* $\mathsf{S} = 0$ *is* **random**. *In particular, a sequential stream with* $\mathsf{S} = 1$ *is* completely *sequential while a stream with* $0 < \mathsf{S} < 1$ *is* partly *sequential.*

If a stream is sufficiently long, then the sequentiality of the stream, S, is the probability that the next request generated is sequential. Each request in a stream $< r, [r^*|s^*]^* >$ can be viewed as an outcome of an experimental trial, where a sequential request indicates "success" whereas a random request indicates "failure". Assume that each trial is independent of other trials. Synthetic I/O stream workloads in storage system simulators like Disksim [25] are generated using this assumption.

By definition, a request is sequential only if it is contiguous to the previous request. It cannot be determined whether the previous request is random or sequential without considering the entire history. By the independence assumption, history is irrelevant, so without loss of generality, let the previous request be the first request and set it to random. Then,

the sub-sequence <r,s> is obtained with probability S,

the sub-sequence <r, r, s> is obtained with probability R× S,

and the sub-sequence $< \underbrace{r, r, ..., r}_{l}, s >$ is obtained with probability $\mathsf{R}^{l-1} \times \mathsf{S}$.

In other words, the probability that the length of a random run is l is given by $\mathsf{R}^{l-1} \times \mathsf{S}$. This is the probability mass function of the geometric random variable [99]. Thus, viewing a stream as an independent sequence of sequential and random requests, the length of a random (sequential) run has a geometric distribution, and it follows that:

the expected (average) length of a sequential run is given by 1/R;

the expected length of a random run is given by 1/S;

the expected length of a run is given by $1/(\mathsf{S} \times \mathsf{R})$;

the expected number of runs in a stream with n requests is $n \times \mathsf{S} \times \mathsf{R}$.

We now compute the maximum prefetch hit rate per stream, for each of the prefetching schemes. The computations take into account that the requests from a stream are interleaved with requests from other streams. Therefore, 2 sequential requests from a sequential run may not arrive contiguously at the storage system.

Maximum stream hit rate of PA: Regardless of the number of streams in the interleaved I/O workload, every sequential request in each stream is prefetched. As assumed at the start of this section, each sequential prefetched request gets a hit. Thus, for a given stream:

$$\mathsf{H}_{maxPA} = \mathsf{S}$$

Maximum stream hit rate of PoH: The PoH technique only prefetches identified sequential requests from all streams. The first sequential request in a sequential run always misses in the prefetch cache. This missed request triggers the sequential detection module which identifies that the request is the start of a sequential run (by comparing request addresses). Thus, all requests in a sequential run, except for the first sequential request, hit in the cache. The expected number of sequential runs in

a stream with n requests is $n * \mathsf{S} * \mathsf{R}$. Therefore, the expected maximum hit rate is $(\mathsf{S} - \mathsf{S} \times \mathsf{R})$. Thus, for a given stream:

$$\mathsf{H}_{maxPoH} = \mathsf{S}^2$$

Maximum stream hit rate of PoM: Regardless of the number of streams in the interleaved workload, the PoM technique prefetches on a cache miss. If a sequential run consists of an even number of requests, then half the requests hit in the cache. If a sequential run consists of an odd number of requests, then the ceiling of half the requests hit in the cache. This gives:

$\mathsf{S}/2$: hit rate of even length sequential runs

$\mathsf{S}(1 + \mathsf{R})/2$: hit rate of odd length sequential runs

$1/(1 + \mathsf{S})$: probability of obtaining an odd length sequential run

$$\mathsf{H}_{maxPoM} = \frac{\mathsf{S}}{1 + \mathsf{S}}$$

I/O workload maximum hit rate: The prefetch hit rate of a workload of interleaved streams is the average of the hit rate of the M streams in the workload. The maximum hit rate is achieved when all the s requests in all sequential streams hit in the prefetch cache. Therefore, the maximum hit rate is achieved when a scheme such as PA is used. The PoH and PoM schemes do not prefetch all the s requests and as a result cannot achieve the maximum hit rate for the workload. For example, if all the streams in the workload are completely sequential, then H_{maxPA} is 1; if all the streams in the workload are completely random then H_{maxPA} is 0; if all the streams are 50% sequential then H_{maxPA} is 0.5; and if half the streams are completely sequential and the others are random, then H_{maxPA} is 0.5.

In the above discussion, we have omitted several proofs and related results. We report these in an appendix to this chapter.

12.5 Forgoing optimality

Storage systems do not use stream-aware prefetch cache replacement policies, and as a result, the maximum workload prefetch hit rate cannot be achieved with an optimum size prefetch cache, regardless of whether PA, PoM, or PoH is used. Here, we analyze the hit rate achievable for a prefetching technique using a non stream aware prefetch cache replacement policy for a given cache size.

Suppose there are M streams in the I/O workload, $\mathsf{M_s}$ of these streams are sequential. The storage system has no knowledge of streams and therefore is not privy to information on how the streams are interleaved. Therefore, as far as the storage system is concerned, there is equal probability that the next request is from any one of the M streams. In other words, there is a probability of $\frac{1}{M}$ that the next request is from a stream i. This is equivalent to stating that for the storage system, the M streams are interleaved uniformly. The next theorem computes a lower bound on the hit rate that can be achieved by a storage system prefetching technique with an optimal size cache.

Theorem 11. *Let H_{max} represent the maximum hit rate that can be achieved by a prefetching technique with a sufficiently large cache. For a cache employing a LRU/FIFO based eviction policy, the hit rate that can be achieved using an optimal sized prefetch cache is at least $\mathsf{H}_{max}\left(1-\left(\frac{M-1}{M}\right)^{M_s}\right)$, for a workload of uniformly interleaved streams, where some of the streams may be partly sequential.*

Proof. Suppose a sequential prefetch request s_i from stream i is inserted into the cache at the MRU head. We analyze the scenario when the next on-demand request from a stream j arrives.

With probability $1/\mathsf{M}$, $i = j$ and this on-demand request corresponds to prefetch s_i, so prefetch s_i hits in the prefetch cache.

With probability $\frac{\mathsf{M}-1}{\mathsf{M}}$, the on-demand request is from a stream other than i. If

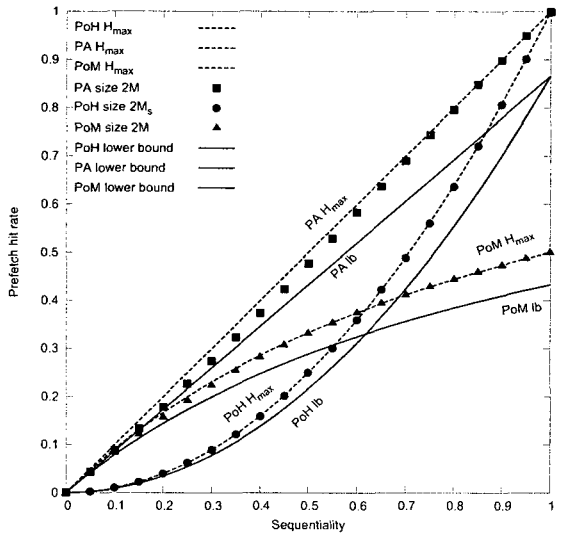


Figure 12-1: Prefetch hit ratio obtained as a function of sequentiality with the prefetch cache size set to 2M, where M=M_s=100 streams of identical sequentiality (X axis).

this on-demand request results in a prefetch cache insertion, then the cache line for s_i either retains its position in the LRU queue or moves 1 line closer to the LRU head. If this on-demand request does not impact the prefetch cache, then the cache line for s_i retains its position in the LRU queue.

Suppose the on-demand request corresponding to s_i does not arrive during the $(M_s - 1)$ on-demand request arrivals after loading s_i at the MRU head. Each arrival will result in s_i either retaining its position in the LRU cache or moving 1 line closer to the LRU head. Since the cache size is M_s, prefetch request s_i is guaranteed to stay in the cache during the next $(M_s - 1)$ on-demand request arrivals. In the worst case, after $(M_s - 1)$ arrivals, the cache line holding s_i would be at the head of the LRU end and would face eviction from the next insertion from a newly identified sequential run.

Thus, for a cache size of M_s, with probability $\left(1 - \left(\frac{M-1}{M}\right)^{M_s}\right)$, a sequential request will hit in prefetch cache within the next M_s insertions. If the maximum hit rate for

the workload that can be achieved by the prefetching technique with a sufficiently large cache is H_{max}, then with an optimal size cache, the prefetching technique would get a minimum hit rate specified in the theorem. □

Theorem 11 gives the minimum expected hit rate for a given workload and a given prefetching technique when the cache size is optimum. Both PA and PoM blindly prefetch requests from the M streams, therefore setting the prefetch cache size to $\mathsf{M_s}$ could result in substantial loss of hits. The cache should be set to a minimum size of M since requests from all M streams are prefetched. On the other hand, PoH only prefetches identified sequential requests from the $\mathsf{M_s}$ streams, so setting the prefetch cache size to $\mathsf{M_s}$ would result in a hit rate close to the maximum. The next Corollary gives the minimum hit rate that each of the techniques can provide for a given cache size.

Corollary 3. *For a cache size of* L *lines, a lower bound on the hit rate is given by:*

$$\begin{aligned}
\mathsf{H}_{PoH} &\geq \mathsf{H}_{maxPoH} \times \left(1 - \left(\frac{M_s - 1}{M_s}\right)^{\mathsf{L}}\right) \\
\mathsf{H}_{PA} &\geq \mathsf{H}_{maxPA} \times \left(1 - \left(\frac{M - 1}{M}\right)^{\mathsf{L}}\right) \\
\mathsf{H}_{PoM} &\geq \mathsf{H}_{maxPoM} \times \left(1 - \left(\frac{M - 1}{M}\right)^{\mathsf{L}}\right)
\end{aligned}$$

Define **optimal size for a scheme** to be the minimum cache size that is needed to get the maximum hit rate for the prefetch scheme when a stream-aware cache replacement policy is used. From Corollary 2 (in Section 12.3) and Corollary 3, it follows that the scheme optimal cache size for PA and PoM is M, while the scheme optimal cache size for PoH is $\mathsf{M_s}$. If the cache size L is set to $k \times \mathsf{M}$ for PA and PoM and $k \times \mathsf{M_s}$ for PoH for some positive integer k, then, $((\mathsf{M} - 1)/\mathsf{M})^{k\mathsf{M}} \rightarrow e^{-k}$ and

$((\mathsf{M_s}-1)/\mathsf{M_s})^{k\mathsf{M_s}} \rightarrow e^{-k}$ as $\mathsf{M}, \mathsf{M_s} \rightarrow \infty$. Thus, for large M and $\mathsf{M_s}$, the hit ratios are bounded as below:

$$\begin{aligned} \mathsf{H}_{PoH} &\geq \mathsf{H}_{maxPoH}\left(1-e^{-k}\right) \\ \mathsf{H}_{PA} &\geq \mathsf{H}_{maxPA}\left(1-e^{-k}\right) \\ \mathsf{H}_{PoM} &\geq \mathsf{H}_{maxPoM}\left(1-e^{-k}\right) \end{aligned}$$

This leads to the following result.

Result 1. *As the prefetch cache size is increased by a constant factor of the optimal cache size for the scheme, the hit rate loss decreases exponentially.*

For example, let M=100, and $\mathsf{M_s}$=50. Consider a completely sequential stream in the workload. The maximum hit rate for this stream is 1. Therefore, if PA is used with a sufficiently large cache, this stream would get a hit rate of 1. By Corollary 3, $\mathsf{H}_{PA} \geq 0.63$ when the prefetch cache size is set to 100, $\mathsf{H}_{PA} \geq 0.87$ when the prefetch cache size is set to 200, and $\mathsf{H}_{PA} \geq 0.95$ when the prefetch cache size is set to 300. Figure 12-1 shows the hit rate provided by the 3 techniques for a cache size set to $2\mathsf{M}$ for PA and PoM, and $2\mathsf{M_s}$ for PoH.

If the workload has a majority of highly sequential streams then a cache size of M ($\mathsf{M_s}$ for PoH) would, on average, give a high hit ratio. For a workload containing partly sequential streams, this is not necessarily true. In fact, even if the number of streams is known to be M, setting the prefetch cache size to that value may not result in optimal performance. When streams have a low sequentiality, the average length of a sequential run in the workload is small, and hence the workload comprises a large number of short sequential runs. The start of a new sequential run results in eviction of data from the LRU end of the cache, because cache replacement schemes, being stream unaware, cannot replace the previously (wrongly) prefetched request

for a stream with the new request for the same stream. Consequently, the hit ratio obtained by a stream unaware prefetching scheme is relatively low, even with an optimal prefetch cache size, when the workload contains partly sequential streams of low sequentiality. Hence, what is required is a prefetching scheme that is able to set the prefetch cache size to a value that is a small constant factor larger than the optimal size, and do so dynamically, without knowledge of the number of streams in the workload. We present such a scheme in the next section.

12.6 Online near-optimal sizing

After each workload request is presented, an online sizing scheme must decide whether to increment, decrement, or leave unchanged the size of the cache. Our results from previous sections show that although setting the prefetch cache size optimally in this way is impossible (see Theorem 10), setting it to a size that is a factor larger than the optimal obtains a prefetch hit rate that is close to the maximum possible for a given workload (see Theorem 11 and Figure 12-1). This, however, apparently requires the knowledge of the number of streams in the workload, which is often a dynamic value and therefore impossible to obtain in advance. It turns out that a simple online sizing scheme does exist that maintains the cache size a small factor larger than the optimal size and obtains a hit rate that is close to the maximum possible for a given workload, with an arbitrarily high (but less than one) probability.

The details of such an online sizing scheme are listed in the pseudocode of Scheme 4. The scheme stores requests that are evicted from the prefetch cache before being hit in the on-demand portion of the read cache. After such an eviction of a request *req*, if a request for *req* arrives in the workload, then the scheme concludes that the prefetch cache must have been smaller than necessary which led to the pre-hit eviction of the prefetched request *req*. The scheme then increases the prefetch cache size and

loads the succeeding prefetched request into this new cache line, anticipating that $req, req + 1, \ldots$ is part of a sequential run.

The sizing scheme cooperates with the caching and prefetching schemes. Whenever a request hits in the on-demand cache (pseudocode lines 4-7), the sizing scheme determines whether this is a non-rereference hit. If so, this means that the request must have been evicted from the prefetch cache. In this case, the scheme concludes that the prefetch cache size must have been smaller than necessary when the request was evicted before being hit. The sizing scheme thus increases the prefetch cache size by one line. Whenever a request hits in the prefetch cache or misses in both the prefetch and the on-demand cache, the sizing scheme leaves the prefetch cache size unchanged.

The actions above guarantee that the prefetch cache size will be incremented in response to addition of new sequential streams or runs. This, however, is not enough: the sizing scheme must also judiciously decrement the cache size when a sequential stream terminates. This is a difficult decision because unlike the decision to increment the cache size where an evicted prefetched request is indicative of cache space scarcity, there is no reliable and timely signal for cache size inflation.

The sizing scheme solves this problem by monitoring the eviction end of the prefetch cache. For example, suppose the prefetch cache size is set to ten cache lines whereas the workload contains only two completely sequential streams. Requests from the two streams will be loaded at the insertion end of the prefetch cache. When these requests are hit, they will be evicted from the prefetch cache and succeeding requests will be loaded into cache lines at the insertion end of the prefetch cache. Eventually, all hits and insertions will be confined to the insertion end of the cache and the eviction end of the cache will see no hits. The sizing scheme exploits the presence of this "quiet zone" at the eviction end of an inflated cache. The scheme

monitors the eviction end of the cache for a sufficiently long period (pseudocode lines 12-16). During this period, if the requests residing near the eviction end do not receive any hits, then the scheme concludes that the cache is inflated and decrements the cache size. The request that is evicted as a result of the cache size reduction is loaded into the on-demand cache. This provides the decrement decision a level of self-correction: if an adequately sized cache is erroneously decremented, then the evicted request will lead back to an increment once the request for the evicted request arrives in the workload. Any decision to decrement the cache size is also ignored if the cache size has been incremented during the monitoring period: this ensures that the decision to increment trumps the decision to decrement because the former is likely to be based on a more reliable indicator.

The monitoring period is set to the sum of current size of the prefetch cache and the size of the on-demand cache. The larger the sum of the on-demand and prefetch cache sizes, the longer is the monitoring period. A longer monitoring period is advantageous because it reduces the chance that an adequate prefetch cache size is erroneously decremented. On the other hand, a long monitoring period also delays the reduction of an inflated cache size to an economical value.

12.7 Simulation results

The performance of the proposed sizing scheme is validated through simulations. We ran the simulations using the CMU Disksim [25] simulator. The simulator is used in slave mode by a caching and sizing module that implemented the PoM, PoH, and PA prefetching schemes and the online sizing scheme. The simulations are carried out using synthetically generated SPC-2-like read workloads [4] and partly sequential workloads. We test the sizing scheme both under uniform and nonuniform interleaving as well as under static and dynamic workloads.

12.7.1 Performance under uniform interleaving

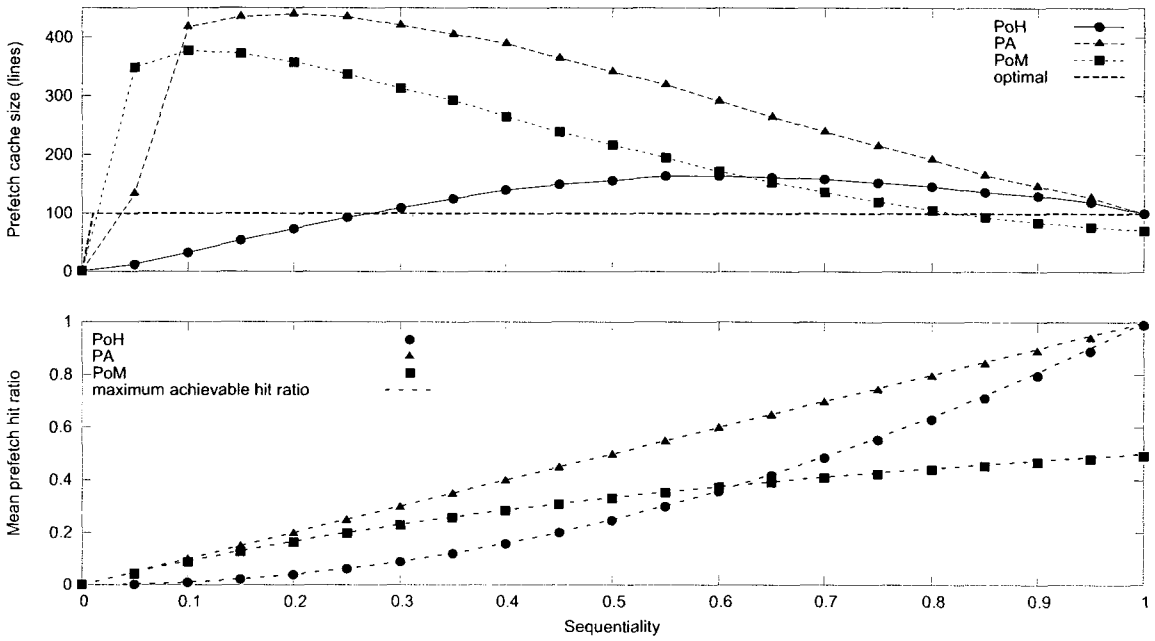


Figure 12-2: The prefetch hit ratio obtained as a function of workload sequentiality with the online prefetch cache sizing scheme listed in Pseudocode 4. The workload contains $M=M_s=100$ streams of identical sequentiality (X axis).

In this experiment, the sizing scheme is subjected to a workload in which streams are interleaved uniformly. The workload contains a total of 100 streams, all of identical sequentiality. The sequentiality is varied from zero (all streams completely random) to one (all streams completely sequential). The hit rate obtained by PA, PoH, and PoM is measured. The maximum prefetch cache size set by each scheme during each simulation run is recorded. Ten such runs are executed and the average of the maximum prefetch cache sizes and the hit rates obtained is computed. The results are plotted in Figure 12-2. The bottom graph in Figure 12-2 shows the hit rates

obtained by the three schemes using the online sizing scheme (solid points) and the maximum theoretical hit rates achievable by those schemes (dashed lines). (These hit rates are derived in Section 12.4.) It is clear that the hit rate obtained by each scheme is within a few percent of the hit rate achievable by that prefetching scheme. This confirms Result 1 derived in Section 12.5.

The top graph in Figure 12-2 shows the maximum prefetch cache size set by the online sizing scheme when it is coupled with each of the three prefetching schemes. The optimal prefetch cache size (dashed horizontal line) is 100 when the number of partly sequential schemes is 100 and zero when all the streams are random (zero sequentiality). PA achieves the highest hit rate of all the schemes but also requires the largest prefetch cache size. This is because PA prefetches succeeding requests blindly. When sequentiality is low, this results in a large number of requests being prefetched into the prefetch cache that are never hit. Nonetheless, these requests end up nudging hitable prefetched requests out of the cache. Such prehit evictions end up in the on-demand cache and hits to them are detected by the sizing scheme which then correctly increments the prefetch cache size. In effect, due to PA's blind prefetching policy, the sizing scheme is forced to maintain a prefetch cache size that is four times larger than the optimal. Yet, this is not a prohibitively heavy price to pay, given that PA achieves close to the maximum achievable hit rate (S, equal to the sequentiality of the workload) of any realizable prefetching scheme on any given workload.

The prefetch cache size set by the sizing scheme for PoM is lower than PA although still close to that used by PA. Like PA, PoM too uses a blind prefetching policy. Thus, for low sequentiality workloads, it ends up paying a heavy price in terms of prefetch cache space as seen in the left portion of Figure 12-2. Moreover, because PoM only prefetches on misses, it fails to obtain hits on almost half of all the hitable

requests. (This is corroborated by the hit rate derived for PoM in Section 12.4: when sequentiality S=1, PoM's hit rate S/(1+S) is only 0.5.) PoM makes up for this suboptimal performance in its reduced prefetching cost. Since PoM piggybacks its prefetch request onto the on-demand request that fetches the requested data, it saves the disk system an additional seek that would have resulted if a separate prefetch request were to be issued. This can result in big savings when the disk is being utilized heavily.

Figure 12-2 also shows that among the three prefetching techniques, PoH, when coupled with the sizing scheme, is able to use the prefetch cache most frugally. This feature is attributable mainly to PoH's detect-and-prefetch approach to prefetching in contrast to PA and PoM's blind prefetching. Because PoH only loads data that belongs to a sequential run with high probability into the prefetch cache, requests evicted from the prefetch cache before being hit are also very reliable indicators of cache space shortage. As a result, the sizing scheme's decisions are more accurate. In contrast, the blind prefetching policies of PA and PoM provide only a noisy indicator of cache adequacy to the sizing scheme.

12.7.2 Nonuniform interleaving

In the first experiment, all streams are of identical sequentiality and contribute the same average number of requests to the workload. In effect, the average number of intervening requests from other streams between two requests from the same stream, is identical for all streams. In this experiment, the sizing scheme is subjected to a workload comprising streams of different rates and sequentialities. The workload contains a total of 100 streams. Of these, 50 streams are completely random (sequentiality zero) and the remaining 50 streams are of identical sequentiality. The sequentiality of these streams is varied from zero to one. The arrival rates of the random streams

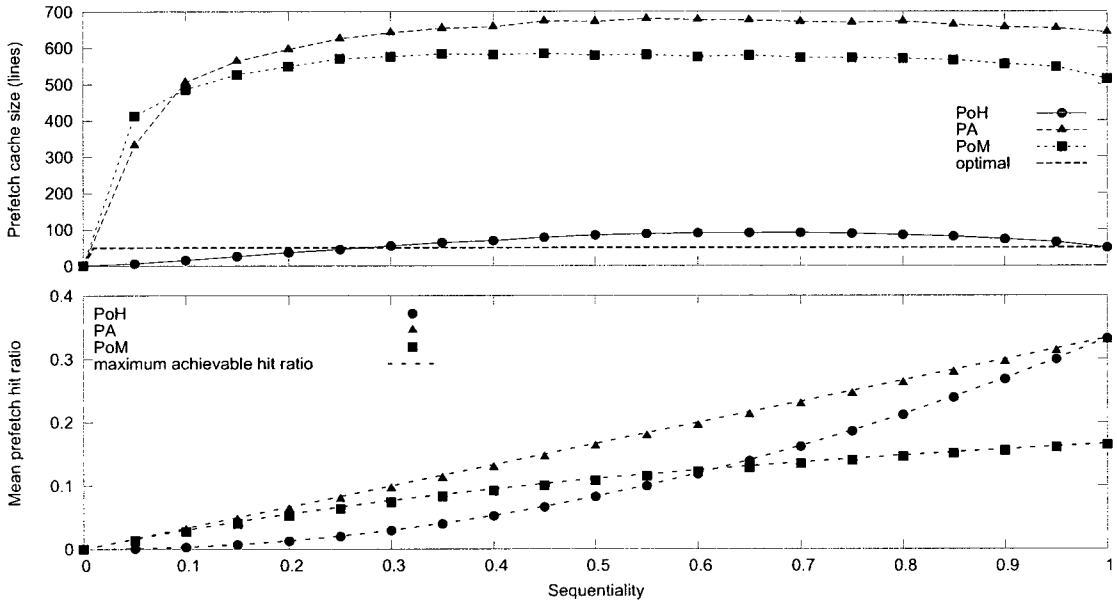


Figure 12-3: The prefetch hit ratio obtained as a function of workload sequentiality with the online prefetch cache sizing scheme listed in Pseudocode 4 under a nonuniform workload. The workload contains 50 completely random and 50 streams of arbitrary (X axis) sequentiality. The arrival rate of the random streams is twice that of the partly sequential streams.

are identical and are set to be twice the arrival rates of the partly sequential streams. The arrival rates of all the partly sequential streams are also set to be identical.

Figure 12-3 shows the hit rates and cache sizes for each of the three prefetching techniques when coupled with the online sizing scheme. Comparing with Figure 12-2 reveals two striking differences. First, the effect of sequentiality on the prefetch cache size is weaker as seen by the relative flatness of the cache size curves. This is readily explained by observing that because half the streams in the workload are completely random and arriving twice as fast as the sequential streams in the workload, the total sequentiality of the workload in the second experiment is much lower than in

the first. As a result, even when the sequential streams are completely sequential (right end of the top graph in Figure 12-3), the prefetch cache size needed is far from optimal, owing to the presence of the high intensity random streams. Second, the hit rates obtained by each of the prefetching schemes are substantially lower than those observed in the first experiment due to the presence of the random streams. The theoretical hit rate obtained by a prefetching scheme on a workload comprising uniformly interleaved streams is defined to be the average of the hit rates obtained by the scheme on each constituent scheme in isolation. When the workload comprises non-uniformly interleaved streams, however, the hit rate of the workload must be generalized appropriately. In this case, the hit rate of the workload is the hit rate obtained by the prefetching scheme on each constituent stream, weighted by its relative intensity.

Figure 12-3 shows that the high intensity random streams have a significant impact on the cache size when the prefetching scheme used is PA or PoM. Since these two schemes prefetch blindly, the high intensity random streams result in a large number of useless prefetches which in turn result in the eviction of a large number of useful prefetches. This forces the sizing scheme to inflate the prefetch cache to a size that can accommodate the useless prefetching of random requests without evicting hitable prefetched requests. In summary, the online sizing scheme is able to deliver near-optimal hit rates even under this relatively challenging workload.

12.7.3 Sizing under a dynamic workload

In this next experiment, a completely dynamic workload is used. While arrival rates and sequentialities of the streams are held constant, unlike experiments in the previous sections, the number of active streams is allowed to vary arbitrarily. The starting time of each new stream is chosen at random within the experimental interval. A

stream once opened remains active for a fixed duration. Figure 12-4 shows example runs under two such scenarios. In both scenarios, 150 random streams open and persist throughout the duration of the experiment. Another 150 streams are completely sequential in the first scenario (Figure 12-4 left) and partly sequential with sequentiality 0.8 in the second scenario (Figure 12-4 right).

In each graph, we plot the simulation time on the X axis and the request addresses in the workload on the left Y axis. Thus, a request at time t for address a is plotted as a point at (t, a). In each graph, on the right Y axis, we plot the instantaneous prefetch cache size maintained by the online sizing scheme coupled with each of the three prefetching schemes and the instantaneous optimal prefetch cache size.

It is clear from Figure 12-4 that the online sizing scheme is able to adjust the prefetch cache size in response to the changing workload. When coupled with an intelligent prefetching scheme like PoH, the cache size maintained by the scheme is within a small factor of the optimal cache size. As indicated in previous results, the hit rate obtained in both scenarios is also within a few percentage of the maximum hit rate achievable by each prefetching scheme.

12.8 Conclusions

This chapter tackles the space requirements of disk array prefetch caches—an expensive and scarce resource. To our knowledge, this is the first work to systematically address the issues and challenges in determining the optimal size of a storage prefetch cache for dynamic I/O workloads. We explore the tradeoff between optimal sizing versus optimal hit rate and prove that no storage prefetch cache scheme based on a realistic replacement policy can simultaneously achieve optimality along both these dimensions. We then compute the hit rate achievable with a given prefetch cache size for various sequential prefetching techniques. We prove that increasing the cache size

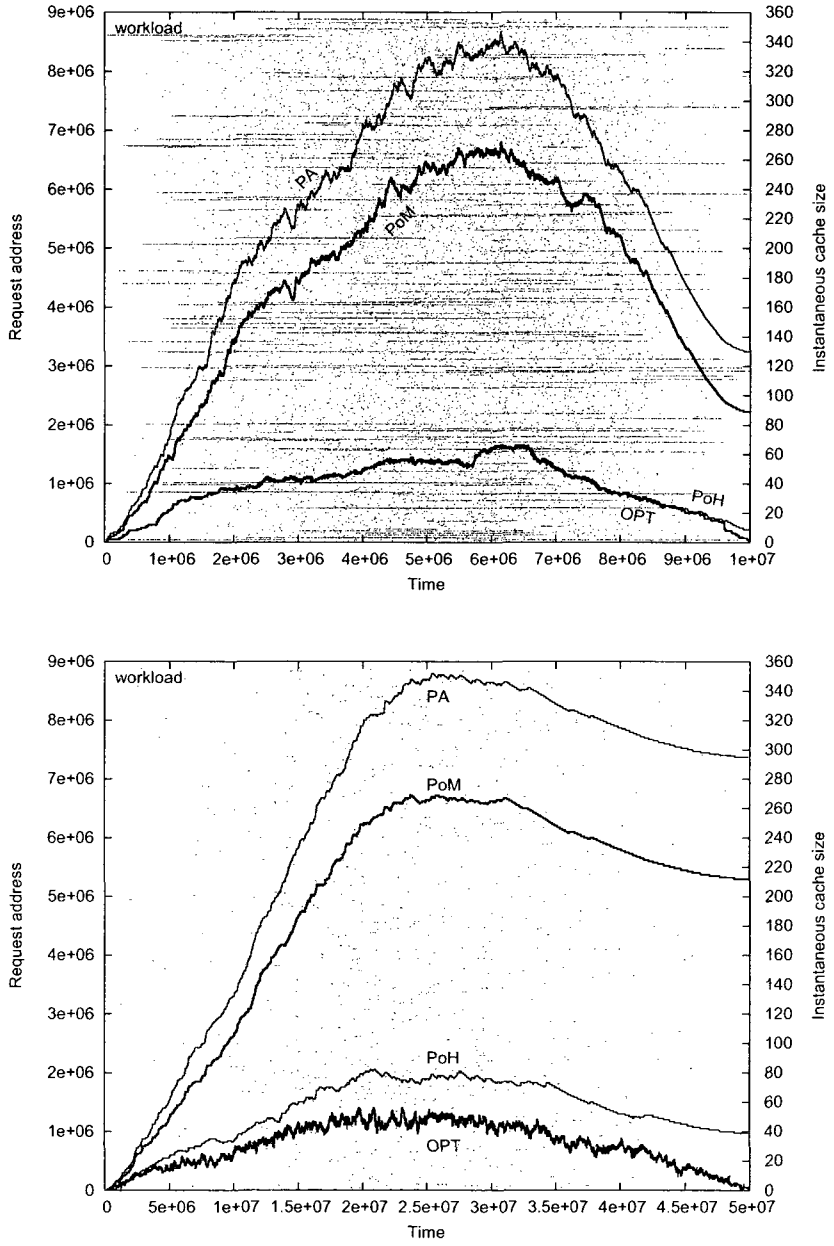


Figure 12-4: An example run of the online sizing scheme on a dynamic workload. *Left:* A total of 150 completely sequential streams open and close dynamically while a background of 150 random streams persists. *Right:* A total of 150 streams of 80% sequentiality open and close dynamically while a background of 150 random streams persists.

beyond the optimal results in exponentially diminishing gains in hit rate. We then propose an online prefetch cache sizing scheme that is able to dynamically maintain the cache size within a small factor of the optimal size while achieving hit rates of over 95% of the maximum hit rate achievable by three popular prefetching schemes. We validate our sizing scheme through simulations with both uniformly and nonuniformly interleaved workloads and static and completely dynamic workloads. Our simulations show that the sizing scheme achieves near-optimal hit rates under all the scenarios tested. The cache size maintained is close to the optimal with the PoH scheme and within a small factor of the optimal with blind prefetching schemes such as PoM and PA.

12.9 Appendix

12.9.1 Expected length of a run

We model the workload and the cache as follows. Unless stated otherwise, all sets and sequences are nonempty. The address space of a storage system is a set $A = \{1, \ldots, s\}$ of nonnegative integers where s is sufficiently large so that $1/s \to 0$. A *request* is an address $r_i \in A$. A *workload* is a sequence of requests $W = \{r_1, r_2, \ldots, r_n\}$. A request r_i is *head-sequential* if and only if $r_{i+1} = r_i + 1$ and *tail-sequential* if and only if $r_i = r_{i-1} + 1$. We shall refer to any contiguous proper subsequence of a workload as a *run* with the length of the run being the number of requests in it. A run $B_{ij} = \{r_i, \ldots, r_j\}, 1 < i \leq j < n$ of a workload $W = \{r_1, \ldots, r_{i-1}, r_i, \ldots, r_j, r_{j+1}, \ldots, r_n\}$ is a *sequential run* if and only if:

1. Request r_i is head-sequential but not tail-sequential,

2. $\forall k : i < k < j$, r_k is head- and tail-sequential, and

3. r_j is tail-sequential but not head-sequential.

A run $B_{ij} = \{r_i, \ldots, r_j\}$ is a *random run* of W if and only if:

1. Request r_{i-1} is tail-sequential but not head-sequential,

2. $\forall k : i \leq k \leq j$, r_k is neither head- nor tail-sequential, and

3. r_{j+1} is head-sequential but not tail-sequential.

Observe that two random runs cannot occur consecutively because by definition they can always be coalesced to form a single random run. Of course, two sequential runs may occur consecutively. Also observe that while a random run of unit length is possible, a sequential run cannot be smaller than two requests. An entire workload is a concatenation of one or more such sequential and random runs of varying length.

Let the probability that request r_i is tail-sequential be p. Then, the probability that request r_i is not tail-sequential will be $1 - p$.

We assume that a workload W is generated by the following algorithm. The first request r_1 is selected uniformly at random from A. Then, for the i-th request ($i > 1$), a biased coin with probability p of turning up heads is flipped. If the coin turns up heads, then request $r_i \leftarrow r_{i-1} + 1$, i.e., r_i is generated to be a tail-sequential request. Otherwise, request r_i is selected uniformly at random from A.

Let us estimate the average length of a sequential run in a workload generated by the above algorithm. We do so by adding some book-keeping variables to the algorithm and then estimating the values of these variables at certain steps of the algorithm. We describe this in further detail below:

1. We add a state variable b to the algorithm which can take values from the set $\{S_1, S_2, R, U\}$. The value held in b denotes the type of request run currently being generated; $b = S_1$ denoting the beginning of a sequential run; $b = S_2$ denoting continuation of a sequential run; $b = R$ denoting generation of a random run and $b = U$ if the type is as yet undetermined.

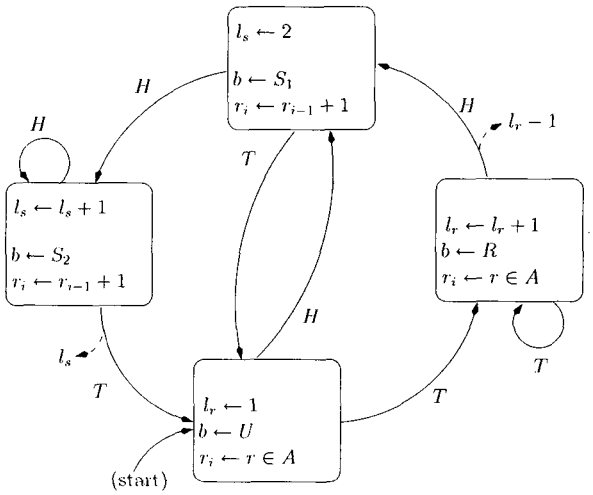


Figure 12-5: A state machine for measuring the length of sequential and random runs.

2. We also add variables l_s and l_r denoting the current length of the sequential or random run (respectively) being generated.

3. Initially, $b \leftarrow U$, r_1 is chosen uniformly at random from A, and $l_r \leftarrow 1$.

4. As before, to generate the next request r_i, a biased coin with probability p of turning up heads is flipped.

5. Depending on the outcome of the coin flip, requests are generated and variables are updated as per the state diagram in Fig. 12-5.

As illustrated in Fig. 12-5, A sequential run begins when $b = S_1$ and the length of the run when $b = S_1$ is $l_s = 2$. Thus, to obtain a sequential run of length $k \geq 2$, an additional $k - 2$ heads are required. A sequential run ends when $b = U$. This is possible only when a final T is obtained. Therefore,

The probability of obtaining a sequential run of length $k \geq 2$, given that a sequential run has begun is

$$\begin{aligned} Pr(l_s = k) &= Pr(k-2 \text{ heads}) \cdot Pr(\text{tail}) \\ &= p^{k-2} \cdot (1-p). \end{aligned} \tag{12.1}$$

Therefore, given a sequential run has begun, its average length is given by:

$$\begin{aligned} E(l_s) &= \sum_{k=2}^{\infty} k \cdot p^{k-2} \cdot (1-p) \\ &= \frac{2-p}{1-p}. \end{aligned} \tag{12.2}$$

The average length of a random run can be obtained similarly.

In order to obtain a random run of length $k \geq 1$, $k-1$ tails are needed with a final head to permit the transition to a sequential run. Therefore, the probability of obtaining a random run of length $k \geq 1$, given that a random run has begin is

$$\begin{aligned} Pr(l_r = k) &= Pr(k-1 \text{ tails}) \cdot Pr(\text{head}) \\ &= (1-p)^{k-1} \cdot p. \end{aligned} \tag{12.3}$$

Therefore, given a random run has begun, its average length is given by

$$\begin{aligned} E(l_r) &= \sum_{k=1}^{\infty} k \cdot (1-p)^{k-1} \cdot p \\ &= \frac{1}{p}. \end{aligned} \tag{12.4}$$

12.9.2 Average number of runs in a workload

Next, we determine the average number of sequential runs in a workload. Every sequential run begins with the pattern "TH", the probability of which is $p(1-p)$. Therefore, a workload of n requests will contain

$$R_s(n,p) = n \cdot p(1-p) \tag{12.5}$$

sequential runs (for large n and $p < 1$). Similarly, every random run begins with the pattern "HTT" (except for the trivial cases at the beginning of the workload or when there is only one random run). The probability of this event is $p(1-p)^2$. Therefore, a workload of n requests will contain

$$R_r(n,p) = n \cdot p(1-p)^2 \tag{12.6}$$

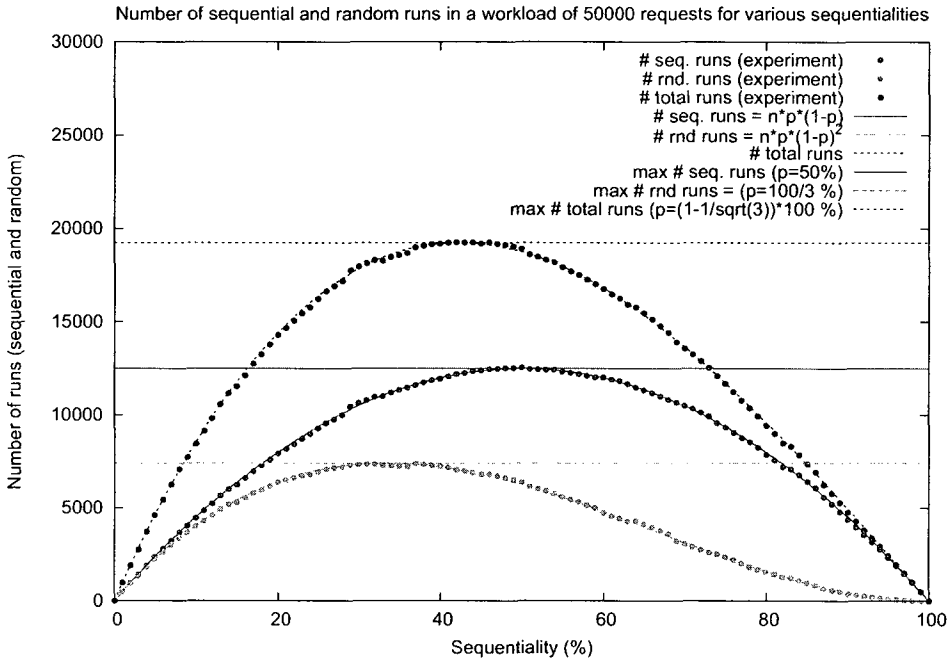


Figure 12-6: Relationship between the number of sequential, random and total runs in any workload as a function of sequentiality.

random runs (for large n and $p > 0$). In all, a workload of n requests will contain a total of

$$R_t(n,p) = n \cdot p(1-p)(2-p) \tag{12.7}$$

runs. We note that $R_s(n,p)$ achieves its maxima at $p = 0.5$ implying that the maximum number of sequential runs in any workload of length n is no more than 25% of the length of the workload (for large n). $R_r(n,p)$ achieves its maxima at $p = \frac{1}{3}$ implying that the maximum number of random runs in any workload of length n is no more than roughly 15% of the length of the workload (for large n). Finally, the total number of runs $R_t(n,p)$ achieves its maxima at $p = 1 - \frac{1}{\sqrt{3}}$ implying that the total number of runs in any workload of length n is no more than roughly 39% of its length. Figure (12-6) illustrates these relationships.

12.9.3 Single stream hit ratio analysis

We ignore random hits and re-reference hits. To simplify the exposition, we first assume that the cache and SSD table size is sufficiently large and that there is only a single stream.

12.9.4 Prefetch-on-miss strategy

Recall the POM strategy: whenever a request r_i is not found to be in the cache, it is served and its succeeding request $r_i + 1$ is pre-fetched into the cache. However, if request r_i is found in the cache, then it is served from the cache and no pre-fetching is performed. Thus, pre-fetching only occurs after a cache-miss.

On average, there will be $E(l_s|b = S_1)$ sequential requests in a sequential run. Consider the length of a sequential run in a stream of sequentiality p. Suppose the length of this run is even. Then, exactly half of the requests in the sequential run will result in cache hits due to the prefetch-on-miss-only rule. Therefore, in a workload of n requests, for a sequential run of even length, the ratio of cache hits to the total requests will be

$$\begin{aligned} r^e_{pm}(p) &= \frac{1}{n} \cdot \left(\frac{1}{2} \cdot n \cdot p(1-p) \cdot \frac{2-p}{1-p} \right) \\ &= p - \frac{p^2}{2}. \end{aligned} \tag{12.8}$$

On the other hand, if the length $E(l_s|b = S_1)$ of the sequential run is odd, then the fraction of requests resulting in a hit will be

$$\begin{aligned} r^o_{pm}(p) &= \frac{1}{2} \cdot (E(l_s|b = S_1) - 1) \cdot p(1-p) \\ &= \frac{p}{2}. \end{aligned} \tag{12.9}$$

The exact value of the average hit ratio of the PM algorithm depends on whether, on an average, a sequential run is of odd or even length. The probability of obtaining a

sequential run of even length is

$$\begin{aligned} Pr(even(l_s)|b = S_1) &= \sum_{i=1}^{\infty} p^{2i-2}(1-p) \\ &= \frac{1}{1+p}. \end{aligned} \tag{12.10}$$

and the probability of obtaining a sequential run of odd length is

$$\begin{aligned} Pr(odd(l_s)|b = S_1) &= \sum_{i=1}^{\infty} p^{2i-1}(1-p) \\ &= \frac{p}{1+p}. \end{aligned} \tag{12.11}$$

The average hit ratio of PM is then the average between Eqn. (12.8) and (12.9) which is

$$\begin{aligned} r_{pm}(p) &= p_{even} \cdot r^e_{pm}(p) + p_{odd} \cdot r^o_{pm}(p) \\ &= \frac{p}{1+p}. \end{aligned} \tag{12.12}$$

12.9.5 TaP strategy

For the TaP strategy, all but the first two requests will result in cache hits. Therefore,

$$\begin{aligned} r_{ssr}(p) &= \frac{1}{n} \cdot \left(n \cdot p(1-p) \cdot \left(\frac{2-p}{1-p} - 2 \right) \right) \\ &= p^2. \end{aligned} \tag{12.13}$$

Comparing eqn. (12.12) and (12.13) shows that the hit ratios of the two algorithms are equal at $p_1 = 0$ and $p_2 = 0.5 \cdot (\sqrt{5} - 1) \approx 0.618034$ with $r_{pm}(p) > r_{ssdp}(p)$ for all $0 \leq p < p_2$. That is, on average, TaP performs better than POM when the sequentiality is greater than about 62%, when the cache is sufficiently large.

12.9.6 Prefetch-always strategy

In this case, all but the first request will be a hit. Therefore, the hit ratio will be

$$\begin{aligned} r_{phm}(p) &= \frac{1}{n} \left(n \cdot p(1-p) \cdot \left(\frac{2-p}{1-p} - 1 \right) \right) \\ &= p. \end{aligned} \tag{12.14}$$

12.9.7 Multiple stream analysis

So far, we have assumed a single stream with probability of sequential access p. What happens if there are $m \geq 1$ streams with sequentialities $p_1, \ldots, p_m$? Observe that even in the presence of $m \geq 1$ streams, a sequential run in any of the m original streams will result in hits even when the m streams are interleaved, assuming the cache and the TaP table is sufficiently large. In case of TaP, once a sequential stream is detected by TaP, it will continue to prefetch requests into the cache and these prefetched requests will result in cache hits just as in the single stream case. The interleaving of m streams does not as much affect the hit ratio as it affects the sizes of the cache and virtual table. The argument is similar for POM. If POM prefetches a request that is part of a sequential stream, then the request will be available in the cache in the interleaved case as well. Therefore, our analysis for the single stream holds for the multistream case as well with the following simple modification:

$$r_{pm}(P_m) = \frac{1}{m} \cdot \sum_{i=1}^{m} \left(\frac{p_i}{1+p_i} \right) \tag{12.15}$$

where P_m is a vector of m sequentialities. Similarly,

$$\begin{aligned} r_{ssdp}(P_m) &= \frac{1}{nm} \cdot \sum_{i=1}^{m} n \cdot p_i(1-p_i) \cdot \left(\frac{2-p_i}{1-p_i} - 2 \right) \\ &= \frac{1}{m} \cdot \sum_{i=1}^{m} p_i^2. \end{aligned} \tag{12.16}$$

12.9.8 Cache and table size

We now derive a simple estimate of the cache and table size required to achieve high hit ratio with the TaP strategy.

Consider a workload consisting of m intervleaved streams and consider the behavior of some prefetching technique $\mathcal{P}$ on it. For optimal performance, any prefetching strategy $\mathcal{P}$ must ensure that a request prefetched for stream j "survives" all the other requests from the $m-1$ other streams. That is, if algorithm $\mathcal{A}$ prefetches a request

into the cache for stream j, then this prefetched request must be available in the cache at least until the next request from stream j is received.

Consider an m-interleaved workload $W = (r_1, \ldots, r_i, r_{i+1}, \ldots, r_{i+k}, r_{i+k+1}, \ldots, r_n)$ and consider a subsequence of W $R_j(i) = (r_i, r_{i+1}, \ldots, r_{i+k}, r_{i+k+1})$ such that requests r_i and r_{i+k+1} belong to stream j and all other requests $r_p, i < p < i + k + 1$ belong to streams other than stream j. $R_j(i)$ is known as a j-recurrent subsequence of W beginning at position i and is said to be of length $L(R_j(i)) = k$ (≥ 0). Let $C(R_j(i))$ denote the number of cache insertions resulting from a j-recurrent subsequence $R_j(i)$ (excluding those due to requests r_i and r_{i+k+1}). Notice that in general, for any prefetching strategy $\mathcal{A}$, $C(R_j(i)) \leq L(R_j(i))$ since any request can result in at most one cache insertion. Consider the worst case when $C(R_j(i)) = L(R_j(i))$. Then, it is easy to show that

$$Pr(L(R_j(i)) = k) = \left(\frac{m-1}{m}\right)^k \cdot \frac{1}{m} \tag{12.17}$$

$$= Pr(C(R_j(i)) = k). \tag{12.18}$$

Now suppose the cache is of size $\mathcal{C} > 0$. In the worst case, suppose a sequential request is prefetched and cached at location 0 of the cache. Clearly, for this cached data to be useful, its corresponding request must arrive within $\mathcal{C}$ or fewer requests. The probability that the number of cache insertions in the worst case are no more than can fit in the cache (without evicting our prefetched sequential request from location 0) can be written as

$$Pr(C(R_j(i)) \leq \mathcal{C} - 1) = \sum_{k=0}^{\mathcal{C}-1} \left(\frac{m-1}{m}\right)^k \cdot \frac{1}{m}$$

$$= 1 - \left(\frac{m-1}{m}\right)^{\mathcal{C}}, \tag{12.19}$$

Therefore,

$$Pr(C(R_j(i)) > \mathcal{C} - 1) = \left(\frac{m-1}{m}\right)^{\mathcal{C}}. \tag{12.20}$$

Equation (12.20) provides the probability that a prefetched request is evicted from the cache before it is used. Since this will reduce the hit ratio of the cache, suppose we wish to bound this probability to some small value $\epsilon > 0$. That is we wish that

$$\left(\frac{m-1}{m}\right)^{C} \leq \epsilon.$$

Assuming equality in the worst case, this implies that the cache size

$$C \approx \frac{\log(\epsilon)}{\log\left(\frac{m-1}{m}\right)}. \tag{12.21}$$

On the other hand, we can interpret the above result to mean that with probability $1-\epsilon$, prefetched requests will survive in the cache until they generate a hit. Thus, for any given cache size C, we can compute the hit ratio $r_{\mathcal{P}}(C)$ for any prefetching strategy as

$$r_{\mathcal{P}}(C) \approx (1-\epsilon) r_{\mathcal{P}}, \tag{12.22}$$

where $r_{\mathcal{P}}$ is the hit ratio of prefetching strategy $\mathcal{P}$ with a sufficiently large cache size. In particular, this means that

$$r_{ssdp}(C) \approx \left(1 - \left(\frac{m-1}{m}\right)^{C}\right) \cdot r_{ssdp}. \tag{12.23}$$

(Recall that the above argument ensures that *every* pair of requests from each stream "survives" in the cache from other streams—not just the sequential requests. Thus, it may be possible to tighten this bound.)

12.9.9 TaP table size

Following an approach analogous to the one taken for the cache size, we can obtain a bound on the table size

$$\mathcal{T} \approx \frac{\log(\epsilon)}{\log\left(\frac{m-1}{m}\right)}. \tag{12.24}$$

12.9.10 Detection loss

Definition 17. *Consider any workload generated by m streams of equal rate, exactly one of which is positively sequential. The* **detection loss**, *denoted* $\mathsf{loss}(m, T)$, *of a table of size T is the expected number (taken over all workloads) of* sequential *requests from the positively sequential stream up to and including the first one that hits in the TAP table.*

Proposition 1. $\mathsf{loss}(m, T) = \left(1 - \left(\frac{m-1}{m}\right)^T\right)^{-1}$.

Proof. Let stream j be the positively sequential stream with sequentiality p and consider its ith request. Let d_i be the number of requests from streams other than j up to and including the $(i+1)$-th request from stream j. Stream j will be detected by a table of size T only when $d_i \leq T$ for the first time. Let $P_{\leq}$ be the probability of this event. The number of sequential requests from stream j missed until such a success is a random variable with a geometric distribution and expectation $1/P_{\leq}$.

Given a complete graph (Markov chain) on m vertices labeled 1 through m, the probability of returning to vertex j in no more than $k \geq 1$ steps is

$$\Pr(d_i \leq k) = \sum_{s=1}^{k} \frac{1}{m} \left(\frac{m-1}{m}\right)^{s-1} = 1 - \left(\frac{m-1}{m}\right)^{k}, \tag{12.25}$$

which is also the cumulative distribution of the i.i.d. d_i. Setting $k = T$ gives $P_{\leq}$. From this, the claim follows. □

Proposition 2. *The complementary cumulative distribution of the length of a sequential run in a single stream of sequentiality $0 \leq p \leq 1$ is $\Pr(l > k) = p^{k-1}$ for $k \geq 2$.*

Proof. Recall from earlier sections that the probability of obtaining a sequential run of length exactly k, given that a sequential run had begun is $(1-p) \cdot p^{k-2}$. From this,

it follows that

$$\Pr(l > k) \;=\; 1 - \sum_{i=2}^{k} (1-p) \cdot p^{k-2} = p^{k-1}.$$

□

Proposition 3. *A workload generated by m streams, exactly one of which is positively sequential with sequentiality* $0 < p < 1$ *under a table of size* T *and a sufficiently large cache will accrue an expected hit ratio (taken over the space of all workloads) of*

$$r \;=\; \left(\sum_{s=2}^{\infty} (1-p) \cdot p^{s-2} \cdot \left(\sum_{k=1}^{s-1} (s-k-1) \cdot (1-q) \cdot q^{k-1} \right) \right) p \cdot (1-p),$$

where $q = \left(\frac{m-1}{m}\right)^T$.

Proof. Let s be the length of a sequential run in the single positively sequential stream i. Then, recall that

$$\Pr(s = k) \;=\; (1-p) \cdot p^{k-2}. \tag{12.26}$$

For a sequential run of length s, let $k+1$ be the detection loss. For this, we require $k-1$ losses (missed detection opportunities) and a table hit on the kth request. Thus,

$$\Pr(l = k+1) \;=\; (1-q) \cdot q^{k-1}, \tag{12.27}$$

where $q = ((m-1)/m)^T$ is the probability of returning to stream i in more than T steps. Thus, in a sequential run of length s, the expected number of hits is

$$\left(\sum_{k=1}^{s-1} (s-(k+1)) \cdot (1-q) \cdot q^{k-1} \right) \tag{12.28}$$

where $k+1$ is the loss. Taking expectation over the possible lengths of a sequential run gives the expected number of hits in a single sequential run as

$$\left(\sum_{s=2}^{\infty} (1-p) \cdot p^{s-2} \cdot \left(\sum_{k=1}^{s-1} (s-k-1) \cdot (1-q) \cdot q^{k-1} \right) \right). \tag{12.29}$$

Since the number of sequential runs in a stream of length n is $p \cdot (1-p) \cdot n$, multiplying by this factor and dividing by n gives the expected hit rate as claimed. □

Remark 1. The infinite series in Proposition 3 does not have a closed form. However, it is easy to see that the probability that the length of a sequential run is no more than L, i.e.,

$$\Pr(s \leq L) = \sum_{j=2}^{L} (1-p) \cdot p^{j-2} \tag{12.30}$$

$$= 1 - p^{L-1}. \tag{12.31}$$

Thus, the probability that the length of a sequential run is larger than L is p^{L-1}. Let this probability be no more than some small positive constant ϵ. Then $L \geq 1 + \log(\epsilon)/\log(p)$.

The probability of getting a sequential run of a length larger than L is an arbitrarily small constant ϵ. Hence, we can approximate the infinite series sum with a finite sum up to L. After some algebraic simplification, this gives the following closed form expression for the hit ratio:

$$r = \left(\sum_{s=2}^{L} (1-p) \cdot p^{s-2} \cdot \left(\sum_{k=1}^{s-1} (s-k-1) \cdot (1-q) \cdot q^{k-1} \right) \right) p \cdot (1-p) \tag{12.32}$$

$$= \left(\frac{p}{(qp-1)(q-1)} \right) \cdot \left(\left(p - p^L L - p^{L+1} + p^{L+1} L\right) q^2 + \left(p^{L-1} L - 2p - p^{L+1} L + 2p^{L+1}\right) q \right.$$

$$\left. + p^{L-1} + 2p^L q^L - p^{L+1} q^L - 2p^L + p^L L - p^{L-1} L - p^{L-1} q^L + p \right) \tag{12.33}$$

where $q = \left(\frac{m-1}{m}\right)^T$, $L = 1 + \frac{\log(\epsilon)}{\log(p)}$, and ϵ is an arbitrarily small positive constant. The hit ratio predicted by this approximation is compared with experimental results in Figure 12-7.

When T is sufficiently large (take $T = \infty$) and $q = 0$, the infinite series from Proposition 3 evaluates to p^2 as expected from our earlier analysis (see Eqn. 12.13).

Proposition 4. *Let K be the number of steps such that the probability of returning to a vertex j of a completely connected m-state Markov chain in at least $K+1$ steps*

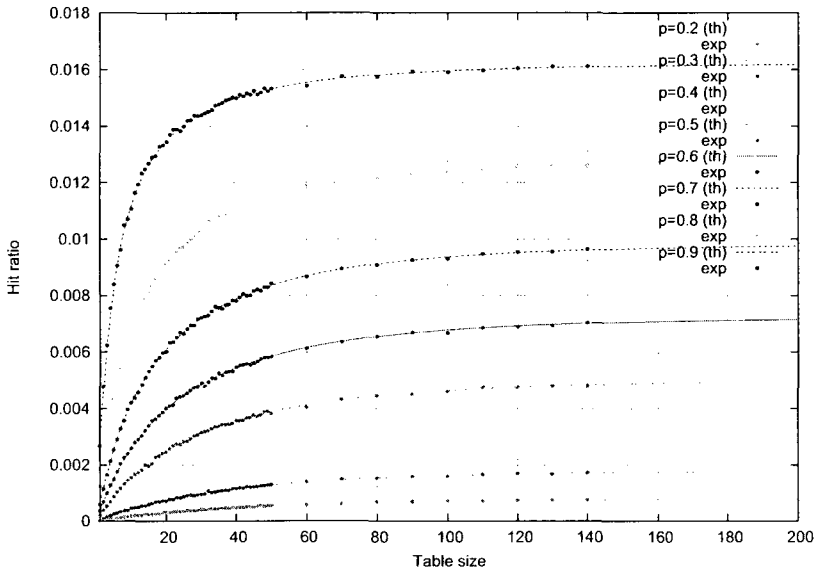


Figure 12-7: The hit rate predicted by the formula in Proposition 3 (the approximate closed form of Eqn. 12.33) is plotted with the hit ratio observed in experiments. The number of streams $m = 100$, the cache is sufficiently large, and $\epsilon = 10^{-6}$.

is at most $\epsilon > 0$. *Then*

$$K \geq \frac{\log(\epsilon)}{\log\left(\frac{m-1}{m}\right)}.$$

Proof. Given a completely connected m-state Markov chain, the probability of returning to vertex j within $K > 0$ steps is

$$\Pr(t \leq K) = \sum_{s=1}^{K} \frac{1}{m}\left(\frac{m-1}{m}\right)^{s-1} = 1 - \left(\frac{m-1}{m}\right)^{K}. \tag{12.34}$$

Since $\Pr(t > K) = 1 - \Pr(t \leq K)$, is the probability of returning to the vertex j in more than K steps, setting $\Pr(t > K) \leq \epsilon$ proves the claim. □

Proposition 5. *Given* $m > 1$ *streams each of sequentiality* $0 \leq p \leq 1$, *the cache size required by TaP to achieve its highest hit ratio of* p^2 *is*

$$\begin{aligned} C &\geq m + (1-p)\cdot p \cdot \frac{\log(\epsilon)}{\log\left(\frac{m-1}{m}\right)}, \text{ if } 0 < p \leq 1, \text{ and} \\ &= 0 \text{ if } p = 0, \end{aligned}$$

with high probability, where $0 < \epsilon \leq 1$ *can be an arbitrarily small positive constant.*

Proof. Clearly, if $p = 0$, the cache is likely to be unused (because we ignore random hits) and therefore a cache of size $C = 0$ is sufficient. If $p = 1$, then each of the m streams is completely sequential. Once any of the m streams is detected, it is beneficial to prefetch and save every subsequent request into the cache. For this, we require at least one slot in the cache for each stream since fewer than m slots will result in lost hits with high probability. Thus, when $p = 1$, $C \geq m$.

Consider m streams each with $0 < p < 1$. Each stream will comprise random and sequential runs. As $p \to 1$, it becomes likely that all m streams produce their individual sequential runs concurrently. In this case, we require at least one slot to prefetch and save requests from the individual sequential run of each stream. Therefore, at least m slots are necessary, i.e., $C \geq m$.

When $p < 1$, every sequential run is of a finite length. Suppose a sequential run in such a stream has been detected and prefetching for that stream is initiated. The last request in such a run will generate a cache hit and this will cause prefetching of the next request. But since the sequential run has ended, the last prefetched request will remain in the cache without being hit and will be evicted only due to the FIFO policy. (Recall that requests that are hit are evicted immediately.) Moreover, such an "orphan request" might cause useful prefetched data to be evicted before it is hit. To protect against such wrongful evictions we require $C > m$ when $0 < p < 1$.

An orphan request is prefetched into the cache when a sequential run in a stream ends. Each sequential run wastes exactly one cache slot due to an orphan request. Suppose a prefetched request r is waiting to be hit and is stored in slot zero of a cache of size C. From earlier discussions (see Eqn.12.21) we know that request r will be hit within the next $n = \log(\epsilon)/\log((m-1)/m)$ requests, with high probability. We also know (see Eqn. 12.5) that the number of sequential runs in n requests is $n \cdot p \cdot (1-p)$. Thus, the number of cache slots wasted due to orphan requests until request r, which

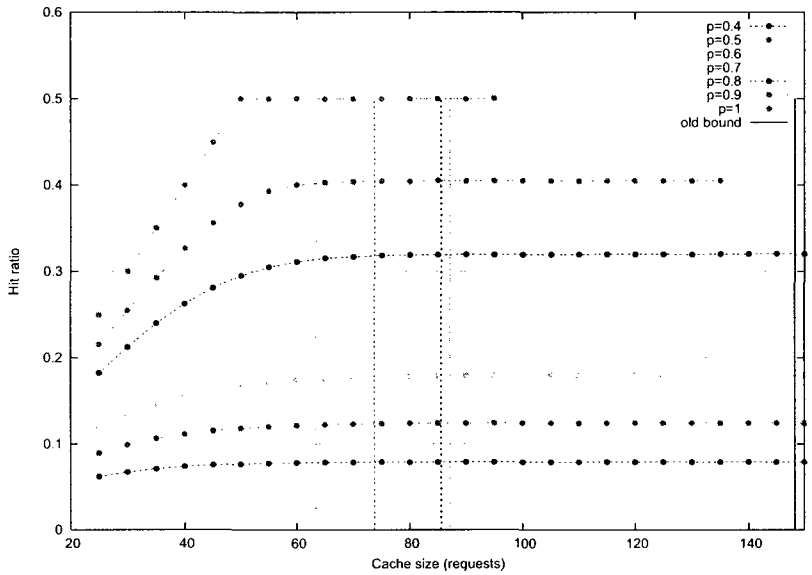


Figure 12-8: Vertical lines show the cache lower bound from Proposition 5. The number of streams $m = 50$, the table is sufficiently large, and $\epsilon = 0.05$. The old cache bound from Eqn. 12.21 is shown by a thick red line.

is the most vulnerable to a wrongful eviction, is hit, is $p \cdot (1-p) \cdot \log(\epsilon) / \log((m-1)/m)$, with high probability. Thus, there will be no wrongful evictions due to orphan requests with high probability if $C \geq m + p \cdot (1-p) \cdot \log(\epsilon) / \log((m-1)/m)$. □

Scheme 4 ONLINE PREFETCH CACHE SIZING

```
1: noEvictionEndHits ← true; noIncr ← true
2: for every request req do
3:     if req is a prefetch cache miss then
4:         if req is a non-rereference hit in the on-demand cache then
5:             Increment prefetch cache size by one line
6:             noIncr ← false
7:         end if
8:     else if req is hit near the eviction end then
9:         noEvictionEndHits ← false
10:    end if
11:    reqCount++
12:    if reqCount == monitoringPeriod then
13:        if noEvictionEndHits and noIncr then
14:            Decrement cache size by one line
15:            Move evicted request into the on-demand cache
16:        end if
17:        reqCount ← 0
18:        noEvictionEndHits ← true; noIncr ← true
19:    end if
20: end for
```

Chapter 13

Hit ratio and response time performance of popular prefetching techniques

The response time of a storage system is an important performance metric. When comparing two caching techniques, the technique that results in a lower storage system response time is more efficient. Since disk service time is an order of magnitude larger than cache service time, a caching technique that has a higher cache hit rate would result in a lower storage system response time. A higher cache hit rate translates into a lower response time for re-reference caching techniques. This rule-of-thumb, however, may not be true for prefetching techniques. It is possible for a prefetching technique with a higher prefetch hit rate to result in a higher storage system response time [75]. The reason for this non-intuitive behavior is that cache hits of prefetching techniques may not be obtained "for free". We illustrate this with an example.

Figure 13-1 shows a workload comprising an increasing number (X axis) of low (10%) sequentiality streams interleaved with (100%) random access streams. The arrival rate of the workload is 100 requests / second, large enough to cause the storage system to be highly utilized (80%-85%). The hit rate data plotted on the left indicates that the AP scheme outperforms all other schemes. The response time data, however, shows that in fact, AP is the worst performer in the group, unless the

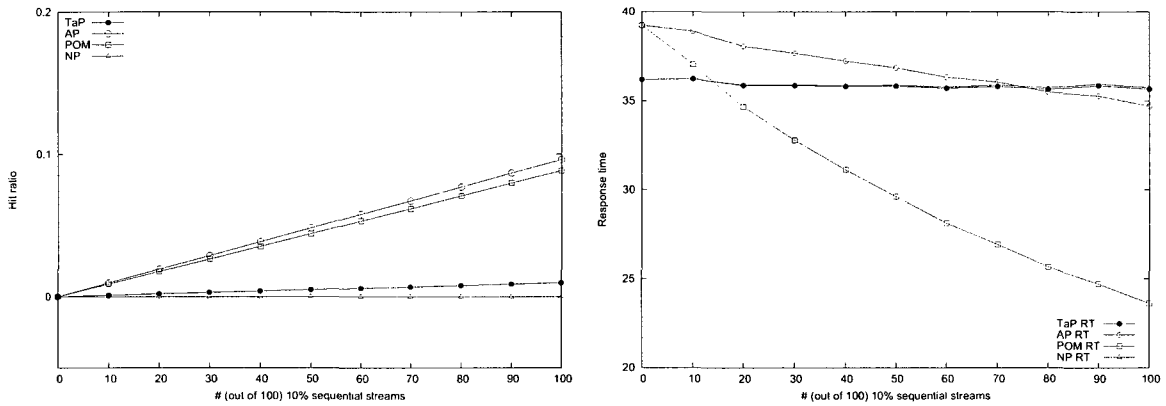


Figure 13-1: Hit rate (left) and response time (right) of a workload containing a varying number (X axis) of 10% sequential streams arriving at a highly utilized (80%-85%) storage system.

workload is sufficiently sequential (rightward on the X axis). Figure 13-1 also shows that POM has the best response time although it has a lower hit rate than AP, and that TaP and NP both have a better response time than AP although their hit rates are lower and different from each other.

These observations can be explained as follows. The AP scheme, by definition, aggressively prefetches on every request, whether it is a cache hit or a miss. Each hit results in a prefetch request for succeeding blocks of data whereas each miss results in a combined fetch-and-prefetch request of double the length for the missed request and its succeeding blocks. While prefetching on cache hits may benefit AP's response time, prefetching on cache misses is guaranteed to be wasteful. Since the workload in this example comprises an overwhelming majority of random requests, and since the storage system is highly utilized, the harm done by AP's wasteful prefetching is too severe to be outweighed by its perfect hit rate. Specifically, AP's response time rides significantly above that of NP because of the time wasted in transferring the succeeding blocks of every random request. Indeed, if the definition of AP is modified

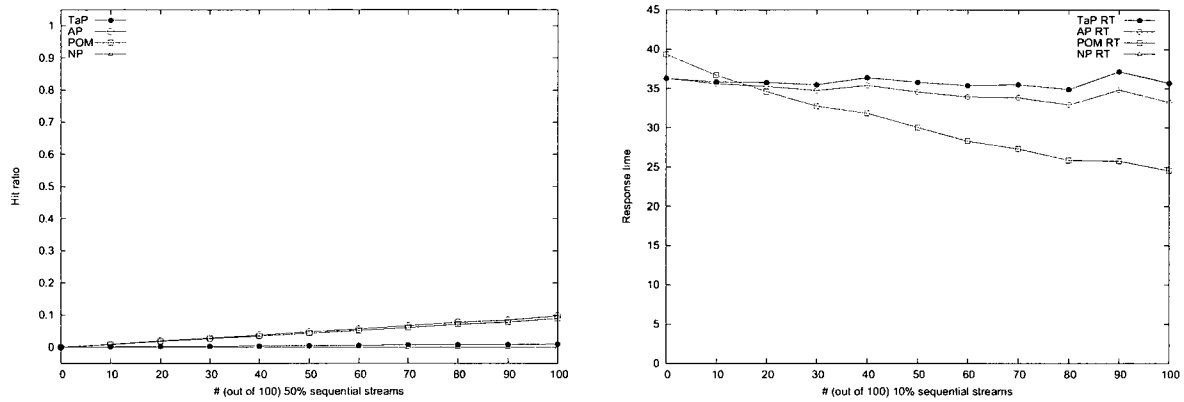


Figure 13-2: Performance of AP when it is modified to never prefetch on a miss due to a random request.

so that AP never prefetches on a miss for a random request (thus eliminating all wasteful prefetches triggered by random requests), then it's response time improves to become better than NP, as shown in Figure 13-2. Thus, while AP's greedy prefetching does guarantee the best possible hit rate achieveable, this optimality does not come for free and is detrimental to its response time—arguably a more pertinent measure of performance.

Furthermore, compare the response time of AP and POM in Figure 13-1. We recall that for random requests in the workload, AP's behavior is identical to that of POM: a miss triggers a request of double the length requesting the missing data and prefetching the succeeding data. The low sequentiality workload in this example contains a majority of random requests and AP and POM—the worst and the best performers according to response time—behave identically on this portion of the workload. Yet, their response time is vastly different. The wasteful prefetching of random requests cannot wholly explain this difference. Rather, it is their behavior on hits that differentiates their response times. POM piggybacks its prefetch request on a regular fetch triggered by a miss whereas AP creates a separate prefetch request

for every hit. The latter has the dual effect of increasing the number of requests generated by AP relative to POM and the hits obtained by AP being obtained at an extra seek time cost per prefetch triggered by a hit. While the workload has low sequentiality and thus the fraction of obtainable hits for any scheme is small, under high utilization, even this small penalty is amplified enough to manifest itself in the response time. (Indeed, AP's penalty disappears if a similar experiment is performed under lower utilization.) Thus, POM's less costly hits alleviate the harm done by its wasteful prefetching of random requests, whereas AP's costly hits come up short in reversing the damage. (If POM is modified so that its requests comprise two separate requests—a fetch for the data requested and a separate request for prefetching succeeding data—then its response time worsens above all the schemes, as shown in Figure 13-3.) Therefore, AP's time rises above that of NP, whereas POM, despite its identical wasteful prefetching of random requests, has the best response time. Thus, somewhat counterintuitively, AP's increased response time is not only due to its wasteful prefetching of random requests, but also due to the cost of its hits (although its hit rate is the best possible).

We briefly cite another example. Figure 13-4 shows the hit ratio (left) and response time measurements from an experiment in which the workload contained a single completely sequential stream interleaved with a varying number of completely random streams. AP and TaP have identical hit rates on this workload. Their response times, however, differ, with AP's (and POM's) response time being larger than TaP. The explanation for AP's higher response time from the previous example applies to this case too. POM's piggybacked prefetching advantage, however, is not sufficient in lowering its response time in this scenario, except when the workload is lightly random. As more randomness is injected into the workload, the advantage of POM's piggybacked prefetching fades away, and the impact of its wasteful prefetch-

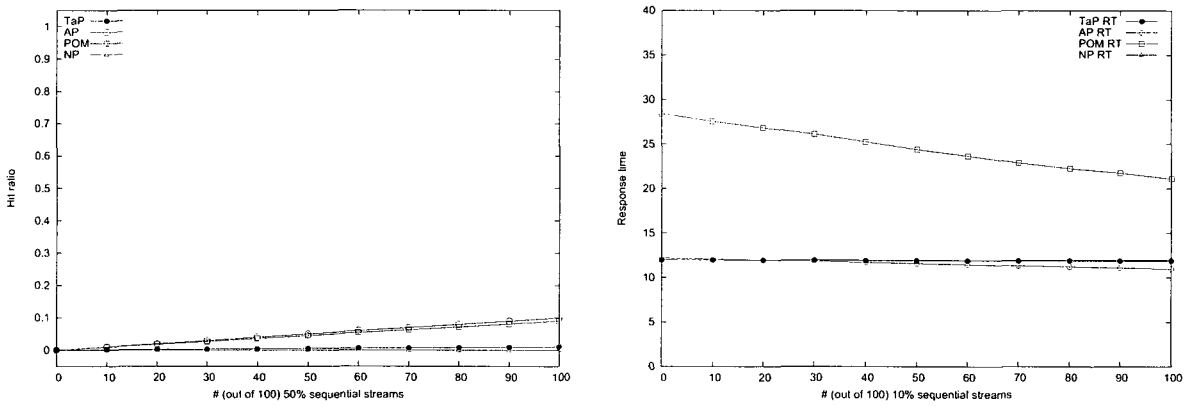


Figure 13-3: Performance of POM when it is modified to send separate fetch and prefetch requests for every miss. (Note: The figure above shows an experiment identical to that of Figure 13-1, except that the storage system utilization was about 50%. We had to decrease the utilization because the disksim simulator runs into saturation due to the large number of events resulting from POM's separate requests at high utilization.)

ing becomes more prominent. Indeed, if both AP and POM are modified so as to never prefetch on a miss due to a random request, then their response time becomes competitive and lower than TaP and NP as shown in Figure 13-5. Note that the corresponding hit rates remain unchanged.

Figures 13-6 and 13-7 show a scenario where the available prefetch cache is limited. In this experiment, the workload comprises m completely sequential streams and $30 - m$ completely random streams. The cache space available is enough to hold exactly 15 requests. (All the requests are of the same length in all the streams.) In addition, TaP's table size is also severely limited: it is given only a single slot in the table. In Figure 13-6, the system is utilized lightly at about 1-2% utilization. It is clear that hit ratio ranks the schemes by their response time: NP obtains the worst response time, followed by POM and AP, and TaP is the clear winner. In Figure 13-7,

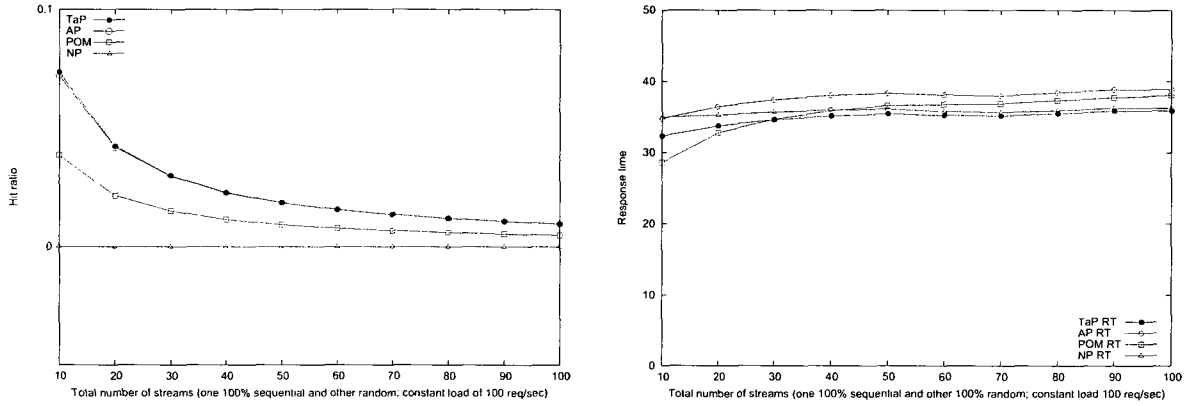


Figure 13-4: Performance of various schemes on a workload containing a single 100% sequential stream and an increasing number of random streams. The storage system utilization was held constant at about 80%.

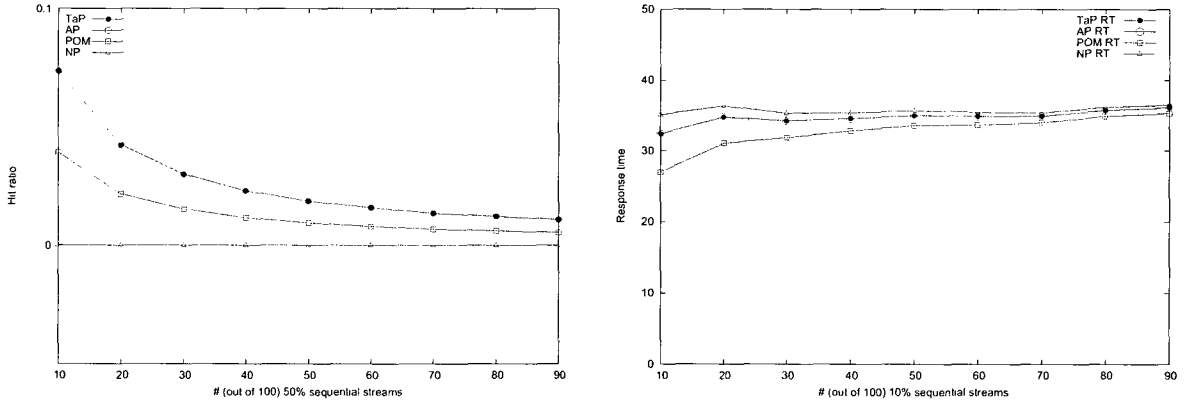


Figure 13-5: When POM and AP are modified to never prefetch on a miss due to a random request and exposed to the same type of workload as in Figure 13-4, their performance improves beyond that of TaP and NP.

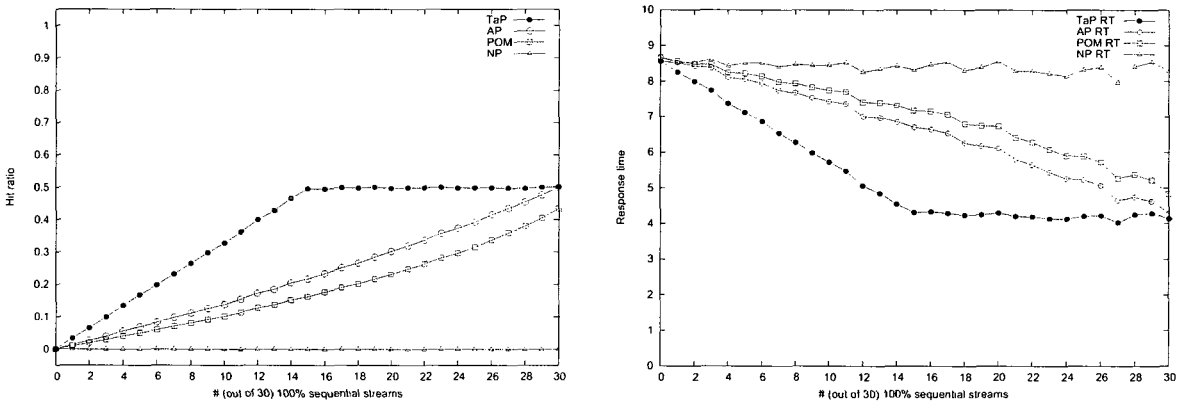


Figure 13-6: Performance of prefetching schemes when the available prefetch cache space is scarce and the storage system is lightly utilized ($\leq 2\%$) by a workload comprising a varying number (X axis) of 100% sequential streams and 100% random streams.

the same experiment is performed under high ($\approx 85\%$) utilization. Here the hit ratio is misleading. AP performs worse than POM although it obtains a better hit rate. Behind this, there are three precipitating factors. AP's wasteful prefetching imposes additional burden on the highly utilized storage system. The wasteful prefetches pollute the cache and result in eviction of hitable data before it is hit. Even without cache pollution, the cache space available is too small to allow AP to obtain its maximal hit rate. While POM faces similar impediments, its piggybacked prefetch request halves the number of sequential requests sent to the disk and thus lowers its overall response time. POM's maximal hit rate is half that of AP and the given cache size is sufficient for POM to obtain it. Figure 13-8 shows performance under a similar experiment with AP modified to not prefetch on a miss when the miss is a random request and POM modified to send separate fetch and prefetch requests. This modification allows AP to eliminate two of the three factors responsible for its poor response time in the previous experiment: wasteful prefetching and cache pollution. As a result, AP matches TaP's hit rate and response time. As for POM, notice that

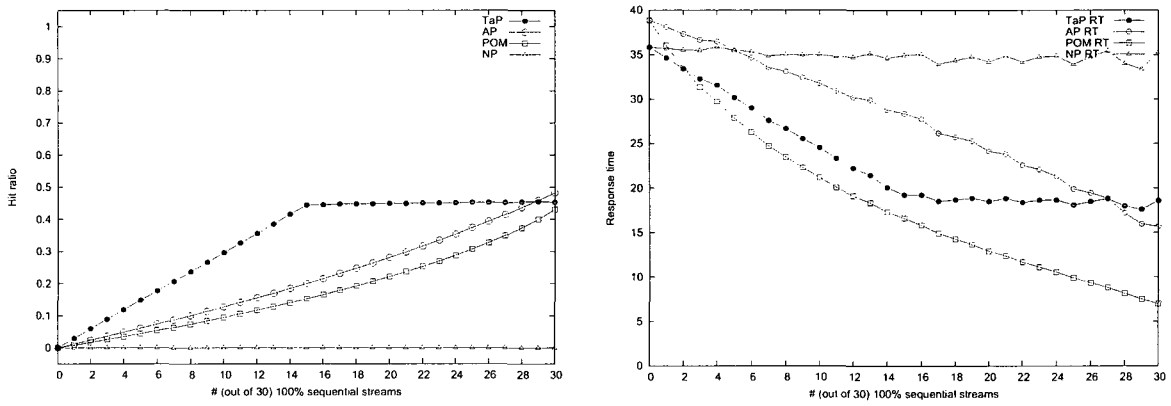


Figure 13-7: Performance of prefetching schemes when the available prefetch cache space is scarce and the storage system is highly utilized ($\approx$ 85%) by a workload comprising a varying number (X axis) of 100% sequential streams and 100% random streams.

POM still almost obtains its maximal hit rate. Yet, as expected, the elimination of its piggybacked prefetch makes its response time the worst in the group. Figure 13-9 shows the same experiment with AP and POM given a sufficiently large cache while TaP is restricted to a small cache as before. (The effect of early eviction and cache pollution is thus eliminated for AP and POM.) AP's response time slowly converges to and then surpasses that of TaP. It is initially higher because AP's wasteful prefetching has an impact on its response time under high utilization and later surpasses that of TaP due to its sufficient cache size. POM's response time is the best because of its piggybacked prefetching advantage. When the workload contains a large amount of sequentiality (right end of the X axis), AP is able to achieve its maximal hit rate which is twice that of POM. As a result, despite POM's piggybacked prefetching and AP's continued wasteful prefetching, AP is still able to beat POM's response time.

The above examples indicate that hit rate alone cannot capture the complex interactions between the composition of the workload, the disk utilization, the specifics of the prefetching heuristic, and the prefetch cache size and its replacement policy.

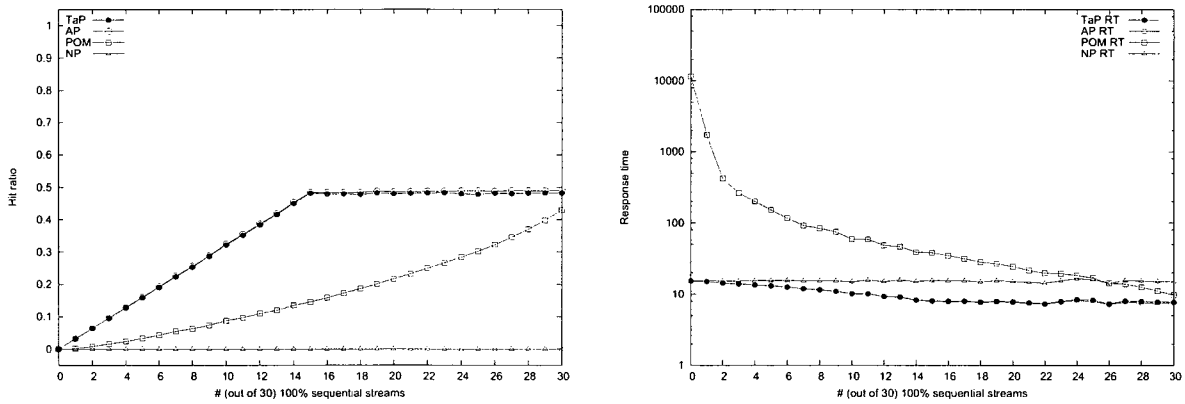


Figure 13-8: Same experiment as reported in Figure 13-7, except AP and POM are modified to not prefetch on a miss when the miss is a random request. (The system was utilized at 70% instead of 85% as in the experiment reported in Figure 13-7. This is because the disksim simulator reaches saturation due to the number of events generated when POM sends separate requests.

A new measure of performance that reflects this complexity is needed. We postpone this to future work.

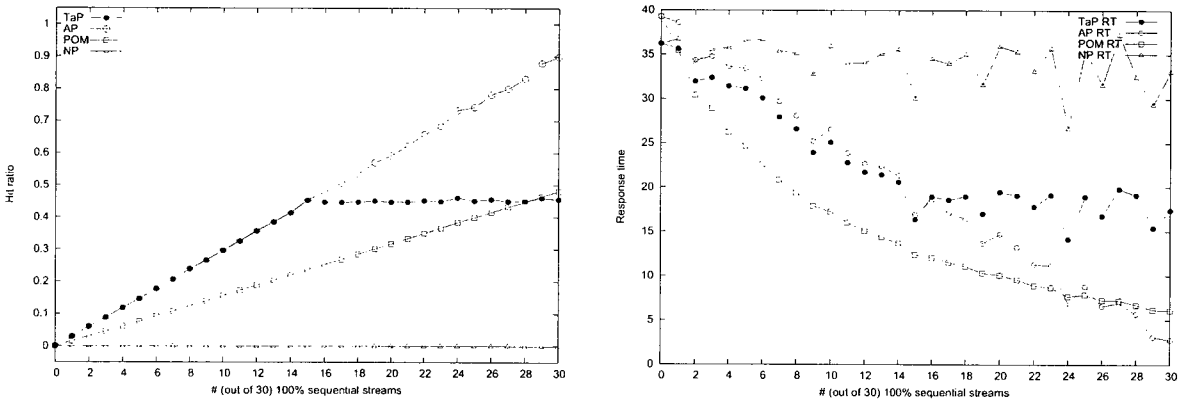


Figure 13-9: Same experiment as reported in Figure 13-7, except AP and POM are provided with a sufficiently large cache.

Bibliography

Bibliography

[1] *Data-Over-Cable Service Interface Specifications-Radio Frequency Interface Specification*, 2000.

[2] *UNH DOCSIS Project*, 2004.

[3] *IEEE Standard for Local and Metropolitan Area Networks, Part 3: CSMA/CD Access Methods and Physical Layer Specifications (802.3ah)*, 2005.

[4] *SPC Benchmark-2(SPC-2) Official Specification, version 1.2.1*, Tech. report, Storage Performance Council, Effective 27 Sept. 2006, `http://www.storageperformance.org/specs`.

[5] Michael O. Albertson and Joan P. Hutchinson, *On the independence ratio of a graph*, Journal of Graph Theory **2** (1978), no. 1, 1–8.

[6] D. Aldous and J. A. Fill, *Reversible Markov Chains and Random Walks on Graphs*, p. 11.

[7] Noga Alon, *Graph powers*, Contemporary Combinatorics (Béla Bollobás, ed.), Bolyai Society Mathematical Studies, Springer, 2000.

[8] Noga Alon and Eyal Lubetzky, *The Shannon capacity of a graph and the independence number of its powers*, IEEE Transactions on Information Theory **52** (2006), no. 5, 2172–2176.

[9] ———, *Independent sets in tensor graph powers*, Journal of Graph Theory **54** (2007), no. 1, 73–87.

[10] Fu-Tai An, Yu-Li Hsueh, Kyeong Soo Kim, Ian M. White, and Leonid G. Kazovsky, *A New Dynamic Bandwidth Allocation Protocol with Quality of Service in Ethernet-based Passive Optical Networks*, IASTED International Conference on Wireless and Optical Communications, July 2003.

[11] W. Anacker and Chu Ping Wang, *Performance evaluation of computing systems with memory hierarchies*, IEEE Transactions on Electronic Computers **16** (1967), no. 6, 764–773.

[12] Dana Angluin, James Aspnes, Zoe Diamadi, Michael Fischer, and Rene Peralta, *Computation in networks of passively mobile finite-state sensors*, Distributed Computing **18** (2006), no. 4, 235–253.

[13] Dana Angluin, James Aspnes, David Eisenstat, and Eric Ruppert, *The computational power of population protocols*, Distributed Computing **20** (2007), no. 4, 279–304.

[14] James Aspnes and Eric Ruppert, *An introduction to population protocols*, Bulletin of the European Association for Theoretical Computer Science **93** (2007), 98–117.

[15] Chadi M. Assi, Yinghua Ye, Sudhir Dixit, and Mohamed A. Ali, *Dynamic Bandwidth Allocation for Quality-of-Service Over Ethernet PONs*, IEEE Journal on Selected Areas in Communications **21** (2003), no. 9, 1467–1477.

[16] Xiaofeng Bai, Abdallah Shami, and Chadi Assi, *Statistical Bandwidth Multiplexing in Ethernet Passive Optical Networks*, IEEE Global Telecommunications Conference (GLOBECOM) (St. Louis, MO, USA), November 2005.

[17] ———, *On the Fairness of Dynamic Bandwidth Allocation Schemes in Ethernet Passive Optical Networks*, Elsevier Journal of Computer Communications **29** (2006), no. 11, 2123–2135.

[18] Xiaofeng Bai, Abdallah Shami, Nasir Ghani, and Chadi Assi, *A Hybrid Granting Algorithm for QoS Support in Ethernet Passive Optical Networks*, IEEE International Conference on Communications (ICC), 2005, pp. 1869–1873.

[19] R. Bartoš, C. K. Godsay, and S. Fulton, *Experimental Evaluation of DOCSIS 1.1 Upstream Performance*, Proc. of the IASTED International Conference on Parallel and Distributed Computing and Networks (PDCN) (Innsbruck, Austria), February 2004.

[20] Vaclav E. Beneš, *Mathematical theory of connecting networks and telephone traffic*, Academic Press, 1965.

[21] Dimitri Bertsekas and Robert Gallagher, *Data Networks*, Prentice Hall, 1992.

[22] S. Bhatia and R. Bartoš, *Performance of the IEEE 802.3 EPON Registration Scheme Under High Load*, Proc. of the SPIE Optics East Conference on Performance, Quality of Service and Control of Next-Generation Communication Networks II (Philadelphia, USA), October 2004.

[23] Swapnil Bhatia, *Design of DBA Algorithms for EPONs (abstract)*, IEEE Conference on Computer Communications (INFOCOM 2006) (Barcelona, Spain), April 2006.

[24] Swapnil Bhatia, Dmitri Garbuzov, and Radim Bartoš, *Analysis of the Gated IPACT Scheme for EPONs*, IEEE International Conference on Communications (ICC 2006) (Istanbul, Turkey), June 2006.

[25] John S. Bucy and G. R. Ganger, *The DiskSim simulation environment version 4.0 reference manual*, Tech. Report CMU-PDL-08-101, Carnegie Mellon University, School of Computer Science, May 2008.

[26] Hee Jung Byun, Ji Myung Nho, and Jong Tae Lim, *Dynamic Bandwidth Allocation Algorithm in Ethernet Passive Optical Networks*, Electronics Letters **39** (2003), no. 13, 1001–1002.

[27] Pei Cao, Edward W. Felten, Anna R. Karlin, and Kai Li, *A study of integrated prefetching and caching strategies*, SIGMETRICS '95/PERFORMANCE '95: Proceedings of the 1995 ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems, ACM Press, 1995, pp. 188–197.

[28] Paolo Casari, Milica Stojanovic, and Michele Zorzi, *Exploiting the bandwidth-distance relationship in underwater acoustic networks*, IEEE/MTS Oceans 2007 (Vancouver, BC, Canada), October 2007, pp. 1–6.

[29] G. Chandrasekaran, M. Hawa, and D. W. Petr, *Preliminary Performance Evaluation of QoS in DOCSIS 1.1*, Tech. Report ITTC-FY2003-TR-22736-01, Information and Telecommunication Technology Center, University of Kansas, January 2003.

[30] K. Mani Chandy and Michel Charpentier, *Self-similar algorithms for dynamic distributed systems*, 27th International Conference on Distributed Computing Systems (ICDCS'2007), June 2007.

[31] Ioannis Chatzigiannakis, *Design and analysis of distributed algorithms for basic communication in ad-hoc mobile networks*, Computer science and engineering,

Dept. of Computer Engineering and Informatics, University of Patras, May 2003.

[32] Jiajia Chen, Biao Chen, and Sailing He, *A Novel Algorithm for Intra-ONU Bandwidth Allocation in Ethernet Passive Optical Networks*, IEEE Communications Letters **9** (2005), no. 9, 850–852.

[33] Sung-Hyun Cho, Jae-Hyun Kim, and Sung-Han Park, *Performance Evaluation of the DOCSIS 1.1 MAC Protocol According to the Structure of a MAP Message*, Proceeding of IEEE ICC 2001 (Helsinki, Finland), June 2001.

[34] Hoon Choi and Kishore S. Trivedi, *Approximate Performance Models of Polling Systems Using Stochastic Petri Nets*, IEEE Infocom (Florence, Italy), May 1992.

[35] Jike Cui and Mansur. H. Samadzadeh, *A new hybrid approach to exploit localities: LRFU with adaptive prefetching*, ACM SIGMETRICS Performance Evaluation Review **31** (2003), 37–43.

[36] Jun-Hong Cui, Jiejun Kong, M. Gerla, and Shengli Zhou, *The challenges of building mobile underwater wireless networks for aquatic applications*, IEEE Network **20** (2006), no. 3, 12–18.

[37] H. A. David and H. N. Nagaraja, *Order Statistics*, ch. 2, pp. 13–14, John Wiley, 2003.

[38] R. M. de Moraes, H. R. Sadjadpour, and J.J. Garcia-Luna-Aceves, *On Mobility-Capacity-Delay Trade-off in Wireless Ad Hoc Networks*, Proc. of IEEE/ACM MASCOTS (Volendam, The Netherlands), October 2004.

[39] A. Dhaini, C. Assi, A. Shami, and N. Ghani, *Adaptive Fairness Through Intra-ONU Scheduling for Ethernet Passive Optical Networks*, International Conference on Communications (ICC) (Istanbul, Turkey), June 2006.

[40] R. Domdom, B. Espey, M. Goodman, K. Jones, V. Lim, and S. Patek, *A Simulation Analysis of the Initialization of DOCSIS-Compliant Cable Modem Systems*, Systems Engineering Capstone Conference (University of Virginia), 2000.

[41] Marc Farley, *Storage networking fundamentals: An introduction to storage devices, subsystems, applications, management, and filing systems*, Cisco Press, 2004.

[42] Joan Feigenbaum, *Product graphs: Some algorithmic and combinatorial results*, Computer science, Dept. of Computer Science, Stanford University, June 1986.

[43] Chuan Heng Foh, Lachlan Andrew, Elaine Wong, and Moshe Zukerman, *FULL-RCMA: A High Utilization EPON*, IEEE Journal on Selected Areas in Communications **22** (2004), no. 8, 1514–1524.

[44] Nasir Ghani, Abdallah Shami, Chadi Assi, and M. Y. A. Raja, *Intra-ONU Bandwidth Scheduling in Ethernet Passive Optical Networks*, IEEE Communications Letters **8** (2004), no. 11, 683–685.

[45] ———, *Quality of Service in Ethernet Passive Optical Networks*, IEEE Sarnoff Symposium, 2004.

[46] Binny S. Gill and Luis Angel D. Bathen, *AMP: Adaptive multi-stream prefetching in a shared cache*, Proc. of USENIX 2007 Annual Technical Conference, 5th USENIX Conference on File and Storage Technologies, Feb 2007.

[47] Binny S. Gill and Dharmendra S. Modha, *SARC: Sequential prefetching in adaptive replacement cache*, Proc. of USENIX 2005 Annual Technical Conference, 2005, pp. 293–308.

[48] J. D. Gindele, *Buffer block prefetching method*, IBM Tech Disclosure Bull. **20** (July 1977), no. 2, 696 – 697.

[49] Arvind Giridhar and P. R. Kumar, *Computing and communicating functions over sensor networks*, IEEE Journal on Selected Areas in Communications **23** (2005), no. 4, 755–764.

[50] ———, *Towards a theory of in-network computation in wireless sensor networks*, IEEE Communications Magazine **44** (2006), no. 4, 98–107.

[51] Omprakash Gnawali, Ben Greenstein, Ki-Young Jang, August Joki, Jeongyeup Paek, Marcos Vieira, Deborah Estrin, Ramesh Govindan, and Eddie Kohler, *The TENET Architecture for Tiered Sensor Networks*, ACM Conference on Embedded Networked Sensor Systems (Sensys) (Boulder, Colorado), November 2006.

[52] C. K. Godsay and Radim Bartoš, *Experimental Evaluation of DOCSIS 1.1 Upstream Performance*, Tech. Report TR 04-01, Dept. of Computer Science, Univ. of New Hampshire, January 2004.

[53] N. Golmie, F. Mouveaux, , and D. Su, *A Comparison of MAC Protocols for Hybrid Fiber/Coax Networks: IEEE 802.14 vs. MCNS*, Proc. IEEE ICC'99 (Nagoya, Japan), June 1999.

[54] Knuth Stener Grimsrud, James K. Archibald, and Brent E. Nelson, *Multiple prefetch adaptive disk caching*, IEEE Transactions on Knowledge and Data Engineering **5** (1993), no. 1, 88–103.

[55] M. Grossglauser and D. Tse, *Mobility Increases the Capacity of Ad Hoc Wireless Networks*, IEEE/ACM Trans. on Networking **10** (2002), no. 4, 477–486.

[56] Matthais Grossglauser and David Tse, *Mobility increases the capacity of adhoc wireless networks*, IEEE/ACM Transactions on Networking **10** (2002), no. 4, 477–486.

[57] Piyush Gupta and P. R. Kumar, *The capacity of wireless networks*, IEEE Transactions on Information Theory **46** (2000), no. 2, 388–404.

[58] J. A. Hoogeveen and A. P. A. Vestjens, *Optimal On-line Algorithms for Single-Machine Scheduling*, Proc. of the 5th Conference on Integer and Combinatorial Optimization, 1996.

[59] Su il Choi, *Cyclic Polling-Based Dynamic Bandwidth Allocation for Differentiated Classes of Service in Ethernet Passive Optical Networks*, Photonic Network Communications **7** (2004), no. 1, 87–96.

[60] Su il Choi and Jae doo Huh, *Dynamic Bandwidth Allocation Algorithm for Multimedia Services over Ethernet PONs*, ETRI Journal **24** (2002), no. 6, 465–468.

[61] Wilfried Imrich and Sandi Klavžar, *Product graphs: Structure and recognition*, Wiley-Interscience Series in Discrete Mathematics and Optimization, John Wiley and Sons, 2000.

[62] Shengming Jiang and Jing Xie, *A Frame Division Method for Prioritized DBA in EPON*, IEEE Journal on Selected Areas in Communications **24** (2006), no. 4, 83–94.

[63] Ahmed Kamal and Brian Blietz, *A Priority Mechanism for the IEEE 802.3ah EPON*, International Conference on Communications (ICC), 2005.

[64] Scott F. Kaplan, Lyle A. McGeoch, and Megan F. Cole, *Adaptive caching for demand prepaging*, Proceedings of the 3rd International Symposium on Memory Management (Berlin), 2002, pp. 221–232.

[65] Jun-Seog Kim, Hun je Yeon, Seog-Gyu Kim, and Jaiyong Lee, *HUHG: High Utilization and Hybrid Granting algorithm for EPON*, AICT-ICIW '06: Proceedings of the Advanced Int'l Conference on Telecommunications and Int'l Conference on Internet and Web Applications and Services (Washington, DC, USA), IEEE Computer Society, February 2006, p. 49.

[66] Jun-Seog Kim, Hun-Je Yeon, and Jaiyong Lee, *HUHG: High Utilization and Hybrid Granting Algorithm for EPON*, International Conference on Communications (ICC) (Istanbul, Turkey), June 2006.

[67] Hemant Kowshik and P. R. Kumar, *Optimal strategies for computing symmetric boolean functions in collocated networks*, IEEE Information Theory Workshop (Cairo, Egypt), January 2010.

[68] Glen Kramer, *Ethernet Passive Optical Networks*, McGraw-Hill Professional, March, 2005.

[69] Glen Kramer, Amitabha Banerjee, Narendra K. Singhal, Biswanath Mukherjee, Sudhir Dixit, and Yinghua Ye, *Fair Queuing with Service Envelopes (FQSE): A Cousin-Fair Hierarchical Scheduler for Subscriber Access Networks*, IEEE Journal on Selected Areas in Communications **22** (2004), no. 8, 1497–1513.

[70] Glen Kramer, Biswanath Mukherjee, Sudhir Dixit, Yinghua Ye, and Ryan Hirth, *Supporting Differentiated Classes of Service in Ethernet Passive Optical Networks*, Journal of Optical Networking **1** (2002), no. 8, 280–298.

[71] Glen Kramer, Biswanath Mukherjee, and Gerry Pesavento, *Interleaved Polling with Adaptive Cycle time (IPACT): A Dynamic Bandwidth Distribution Scheme in an Optical Access Network*, Photonic Network Communications **4** (2002), no. 1, 89–107.

[72] ———, *IPACT: A Dynamic Protocol for an Ethernet PON (EPON)*, IEEE Communications Magazine **40** (2002), no. 2, 74–80.

[73] W. Leland, Murad Taqqu, W. Willinger, and D. Wilson, *On the Self-Similar Nature of Ethernet Traffic*, IEEE/ACM Trans. on Networking **2** (1994), no. 1, 1–15.

[74] Harry Lewis and Christos H. Papadimitriou, *Elements of the theory of computation*, 2nd ed., Prentice Hall, 1997.

[75] Mingju Li, Elizabeth Varki, Swapnil Bhatia, and Arif Merchant, *TaP: Table-based prefetching for storage caches*, 6th USENIX Conference on File and Storage Technologies (FAST '08), 2008, pp. 81–97.

[76] Zhenmin Li, Zhifeng Chen, Sudarshan M. Srinivasan, and Yuanyuan Zhou, *C-miner: Mining block correlations in storage systems*, Proceedings of the 3th USENIX Conference on File and Storage Technologies (FAST), 2004, pp. 173–186.

[77] Y. D. Lin, C. Y. Huang, and W. M. Yin, *An Investigation into HFC MAC Protocols: Mechanisms, Implementation, and Research Issues*, IEEE Communications Surveys & Tutorials **3** (2000), no. 3.

[78] Laszlo Lovasz, *On the Shannon Capacity of a Graph*, IEEE Transactions on Information Theory **25** (1979), no. 1, 1–7.

[79] Yuanqiu Luo and Nirwan Ansari, *Limited Sharing with Traffic Prediction for Dynamic Bandwidth Allocation and QoS Provisioning over Ethernet Passive Optical Networks*, Journal of Optical Networking **4** (2005), no. 9, 561–572.

[80] Maode Ma, Lishan Liu, and Tee Hiang Cheng, *Adaptive Scheduling for Differentiated Services in an Ethernet Passive Optical Network*, Journal of Optical Networking **4** (2005), no. 10, 661–670.

[81] Maode Ma, Yongqing Zhu, and Tee Hiang Cheng, *A Bandwidth Guaranteed Polling MAC Protocol for Ethernet Passive Optical Networks*, IEEE INFOCOM, 2003.

[82] Michael P. McGarry, Martin Maier, and Martin Reisslein, *An Evolutionary Upgrade for EPONs*, Tech. report, Dept. of Electrical Engineering, Arizona State University, September 2004.

[83] ———, *Ethernet PONs: A Survey of Dynamic Bandwidth Allocation (DBA) Algorithms*, IEEE Communications Magazine **42** (2004), no. 8, s8–s15.

[84] ———, *WDM Ethernet Passive Optical Networks*, IEEE Communications Magazine **44** (2006), no. 2, s18–s25.

[85] Hassan Naser and Hussein Moufteh, *A Fast Class-of-Service Oriented Packet Scheduling Scheme for EPONs*, IEEE Communications Letters **10** (2006), no. 4, 396–398.

[86] ———, *A Joint-ONU Interval-based Dynamic Scheduling Scheme for Ethernet Passive Optical Networks*, IEEE/ACM Transactions on Networking **14** (2006), no. 4, 889–899.

[87] S. Palchaudhari, R. Wagner, R. G. Baraniuk, and D. B. Johnson, *COMPASS: An adaptive sensor network architecture for multi-scale communication*, IEEE Wireless Communications (2008), (submitted).

[88] Christos H. Papadimitriou, *Computational complexity*, Addison Wesley, 1994.

[89] Athanasios Papoulis and S. Unnikrishna Pillai, *Probability, Random Variables and Stochastic Processes*, 4th edition ed., McGraw Hill, 2002.

[90] Abhay Parekh and Robert Gallagher, *A Generalized Processor Sharing Approach to Flow Control in Integrated Services Networks: The Single Node Case*, IEEE/ACM Transactions on Networking **1** (1993), no. 3, 344–357.

[91] Chul Geun Park, Dong Hwan Han, and Bara Kim, *Packet delay analysis of dynamic bandwidth allocation scheme in an ethernet pon*, ICN 2005, LNCS 3420 (Pascal Lorenz and Petre Dini, eds.), Reunion Island, France, April 2005.

[92] R. Gary Parker, *Deterministic Scheduling Theory*, Chapman and Hall, 1995.

[93] Jim Partan, Jim Kurose, and Brian Neil Levine, *A Survey of Practical Issues in Underwater Networks*, WUWNet '06: Proceedings of the 1st ACM international workshop on Underwater networks (Los Angeles, CA), ACM Press, 2006, pp. 17–24.

[94] R. Hugo Patterson, Garth A. Gibson, Eka Ginting, Daniel Stodolsky, and Jim Zelenka, *Informed prefetching and caching*, Proc. SOSP Conf., December, 1995.

[95] Ravi Pendse and Ravi Bhagavathula, *Performance of LRU block replacement algorithm with prefetching*, Proceedings of the 1998 Midwest Symposium on Systems and Circuits, IEEE Computer Society, 1998, pp. 86 –.

[96] Michael Pinedo, *Scheduling: Theory, Algorithms, and Systems*, Prentice Hall, 2002.

[97] K. Pruhs, E. Torng, and J. Sgall, *Online Scheduling*, Handbook of Scheduling: Algorithms, Models and Performance Analysis (Joseph Y. T. Leung, ed.), CRC Press, 2004.

[98] J. A. Rice, *Mathematical Statistics and Data Analysis*, ch. 11, 12, Duxbury Press, 1994.

[99] Sheldon M. Ross, *Probability models for computer science*, Academic Press, San Diego, CA, 2002.

[100] Chris Ruemmler and John Wilkes, *An introduction to disk drive modeling*, IEEE Computer **27** (1994), no. 3, 17–29.

[101] V. Sdralia, C. Smythe, P. Tzerefos, and S. Cvetkovic, *Performance Characterization of the MCNS DOCSIS 1.0 CATV Protocol with Prioritized First Come First Serve Scheduling*, IEEE Trans. Broadcast. **vol. 45 no. 2** (1999), 196–205.

[102] Abdallah Shami, Xiaofeng Bai, Chadi Assi, and Nasir Ghani, *Jitter Performance in Ethernet Passive Optical Networks*, IEEE/OSA Journal of Lightwave Technology **23** (2005), no. 4, 1745–1753.

[103] Abdallah Shami, Xiaofeng Bai, Nasir Ghani, Chadi Assi, and H. Mouftah, *QoS Control Schemes for Two-Stage Ethernet Passive Optical Networks*, IEEE Journal on Selected Areas in Communications **23** (2005), no. 8, 1467–1478.

[104] Claude E. Shannon, *The zero-error capacity of a noisy channel*, IEEE Transactions on Information Theory **2** (1956), no. 3, 8–19.

[105] Moshe Sidi, Hanoch Levy, and Steve Fuhrmann, *A Queueing Network with A Single Cyclically Roving Server*, Queueing Systems **11** (1992), 121–144.

[106] Alan Jay Smith, *Sequential program prefetching in memory hierarchies*, IEEE Computer **11** (1978), no. 12, 7–21.

[107] ———, *Sequentiality and prefetching in database systems*, ACM Transanctions on Database Systems **3** (1978), no. 3, 223–247.

[108] Kyuho Son, Hyungkeun Ryu, Song Chong, and Taewhan Yoo, *Dynamic Bandwidth Allocation Schemes to Improve Utilization under Non-Uniform Traffic in Ethernet Passive Optical Networks*, IEEE International Conference on Communications (ICC) (Paris, France), June 2004.

[109] T. Spyropoulos, K. Psounis, and C. Raghavendra, *Spray and Wait: An Efficient Routing Scheme for Intermittently Connected Mobile Networks*, Proc. of ACM SIGCOMM WDTN-05 (Philadelphia, PA), August 2005.

[110] Thrasyvoulos Spyropoulos, Konstantinos Psounis, and Cauligi Raghavendra, *Efficient routing in intermittently connected mobile networks: The single-copy case*, IEEE/ACM Trans. on Networking **16** (2008), no. 1, 63–76.

[111] Hideaki Takagi, *Anlaysis of Polling Systems*, MIT Press, 1986.

[112] Shuji Tasaka, *Performance Analysis of Multiple Access Protocols*, MIT Press, Computer Systems Series, 1986.

[113] Theodore M. Wong and John Wilkes, *My cache or yours? Making storage more exclusive*, Proceedings of the General Track: 2002 USENIX Annual Technical Conference (Berkeley, CA, USA), USENIX Association, 2002, pp. 161–175.

[114] Bruce L. Worthington, Gregory R. Ganger, and Yale N. Patt, *Scheduling algorithms for modern disk drives*, SIGMETRICS '94: Proceedings of the 1994 ACM SIGMETRICS conference on Measurement and modeling of computer systems, ACM Press, 1994, pp. 241–251.

[115] Jin Xie, Shenming Jiang, and Yuming Jiang, *Dynamic Bandwidth Allocation Scheme for Differentiated Services in EPON*, IEEE Communications Magazine **42** (2004), no. 8, s32–s39.

[116] Yeon-Mo Yang, Byung-Ha Ahn, and Ji-Myong Nho, *Supporting Quality of Service by using Delta Dynamic Bandwidth Allocations in Ethernet Passive Optical Networks*, Journal of Optical Networking **4** (2005), no. 2, 68–81.

[117] Lin Zhang, Eung-Suk An, Chan-Hyun Youn, Hwan-Geun Yeo, and Sunhee Yang, *Dual DEB-GPS Scheduler for Delay-Constraint Applications in Ethernet Passive Optical Networks*, IEICE Trans. on Communications **E86-B** (2003), no. 5, 1585–1590.

[118] W. Zhao, M. Ammar, and E. Zegura, *A message ferrying approach for data delivery in sparse mobile ad hoc networks*, ACM MobiHoc, 2004.

[119] ———, *Controlling the mobility of multiple data transport ferries in a delay-tolerant network*, IEEE INFOCOM, 2005.

[120] Jun Zheng and Hussein Mouftah, *Media Access Control for Ethernet Passive Optical Networks: An Overview*, IEEE Communications Magazine **43** (2005), no. 2, 145–150.

[121] Jun Zheng and Shaoren Zheng, *Dynamic Bandwidth Allocation with High Efficiency for EPONs*, International Conference on Communications (ICC) (Istanbul, Turkey), June 2006.

[122] Yongqing Zhu, Maode Ma, and Tee Hiang Cheng, *An Urgency Fair Queueing Scheduling to Support Differentiated Services in EPONs*, IEEE Global Telecommunications Conference (GLOBECOM) (St. Louis, MO, USA), November 2005.

[123] ———, *Hierarchical Scheduling to Support Differentiated Services in Ethernet Passive Optical Networks*, Computer Networks **50** (2006), no. 3, 350–366.